ProRealTime.com
Il riferimento dei software di borsa online

Manuale di Programmazione

ProBuilder

SOMMARIO

Presentazione del ProBuilder

ProBuilder é il linguaggio di programmazione di ProRealTime. Esso serve a concepire degli indicatori tecnici personalizzati, delle strategie di trading (ProBackTest) o delle ricerche personalizzate (ProScreener). ProBackTest e ProScreener sono oggetto di manuali individuali a cause di determinate specificità di programazione.

Si tratta di un linguaggio di tipo BASIC, di utilizzo molto semplice e esaustivo nelle possibilità offerte.

Potrete costruire i vostri propri programmi che utilizzano le quotazioni di qualsiasi strumento quotato, a partire dagli elementi di base:

- le quotazione d'apertura di ogni barra : Open
- le quotazione di chiusura di ogni barra : Close
- il massimo di ogni barra : High
- il minimo di ogni barra : Low
- la quantità di titoli scambiati : Volume.

Le barre, o candele, sono le rappresentazioni grafiche standard delle quotazioni ricevute in tempo reale. ProRealTime vi offre la possibiltà di personalizzare lo stile ed il tipo di grafico, proponendovi, tra le altre, viste tipo Renko, Kagi, Haikin-Ashi.

L'interprete ProBuilder valuta i dati di ogni barra di prezzo dalla più anziana alla più recente, e esegue la formula sviluppata nel linguaggio per determinare il valore degli indicatori sulla barra in questione.

Gli indicatori sviluppati in ProBuilder possono essere visualizzati all'interno del grafico del prezzo oppure in un grafico individuale, a seconda del tipo di scala utilizzata dall'indicatore.

Durante la lettura del presente manuale, assimilirete i comandi che permettono di programmare in ProBuilder, aiutandovi con una esposizione chiara e degli esempi concreti.

Al termine del presente manuale, troverete un indice che vi permetterà di ritrovare tutte le istruzioni ProBuilder, gli indicatori illustrati e d'altre funzioni che completano l'apprendimento della programmazione in ProRealTime.

I lettori già familiari con la programmazione potranno passare direttamente alla lettura del capitolo II, opure consultare l'indice per ritrovare immediatamente la risposta a quesiti specifici.

Per i lettori a digiuno di programmazione, consigliamo innanzitutto la visione del video "Come creare un indicatore in ProBuilder" e la lettura integrale del presente manuale. Essendo molto concreto e diretto,siamo sicuri che in poco tempo sarete in grado di comprendere e gestire i segreti di ProBuilder.

Vi auguriamo una sicura riuscita e buona lettura !

Capitolo 1: Le nozioni fondamentali

Usare ProBuilder

Supporto veloce per la creazione di indicatori

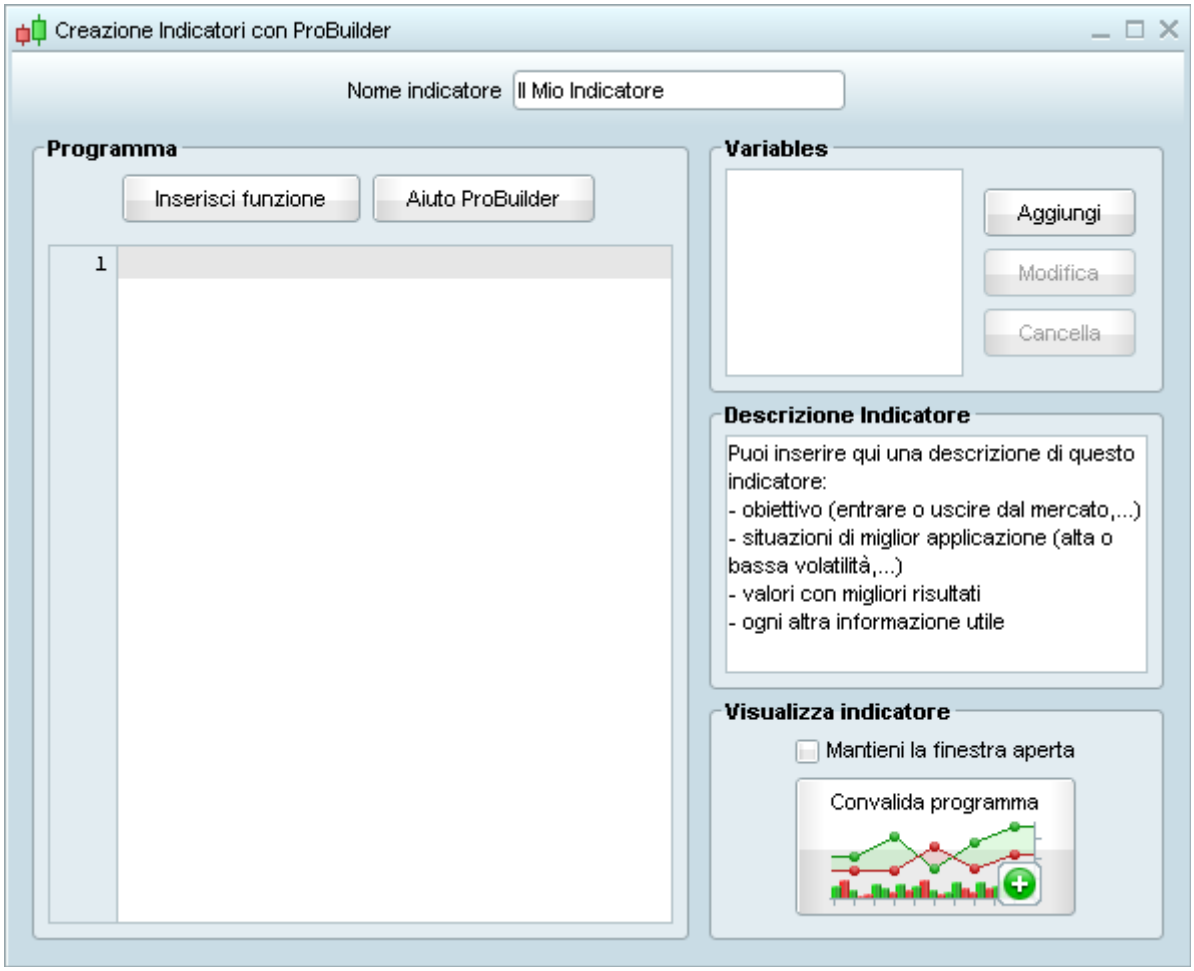
La zona di programmazione di un indicatore é disponibile a partire dal tasto "Indicatori/Backtest" che si trova in alto a destra di ogni grafico della piattaforma ProRealTime.

Accederete dunque alla finestra di gestione degli indicatori, dove potrete:

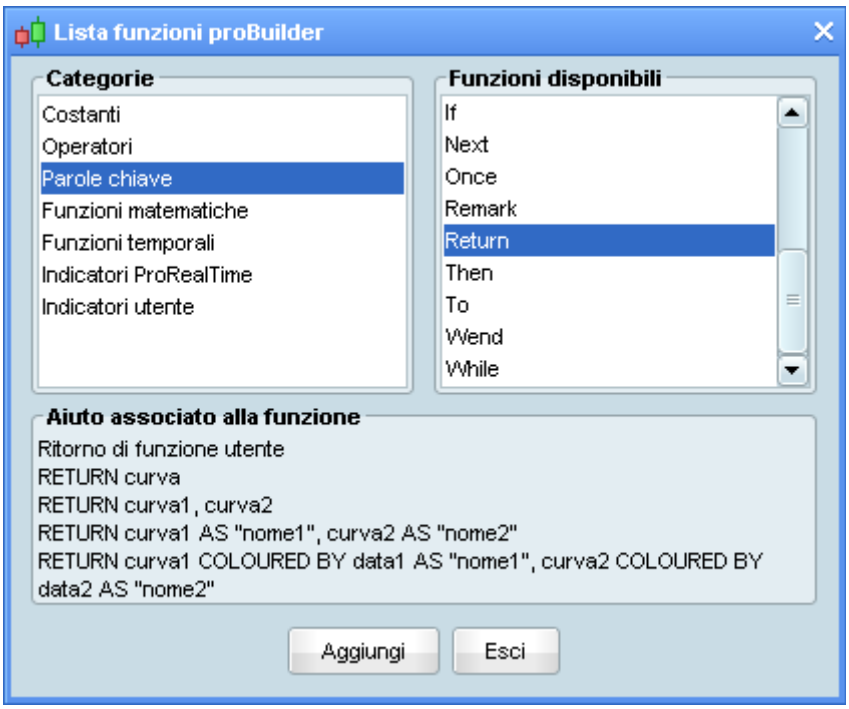
- Visulizzare un indicatore predefinito
- Creare un indicatore personalizzato che potrà in seguito essere applicato a un qualsiasi valore.

In questo secondo caso, cliccate su "Nuovo", per accedere alla finestra di programmazione, che permette di:

- programmare direttemente un indicatore nella zona di testo riservata al codice, oppure
- utilizzare il tasto "Inserisci Funzione", che permette di ritrovare in una nuova finestra la biblioteca delle funzioni disponibili, divise in sette categorie, in modo da permettere una più immediata identificazione.



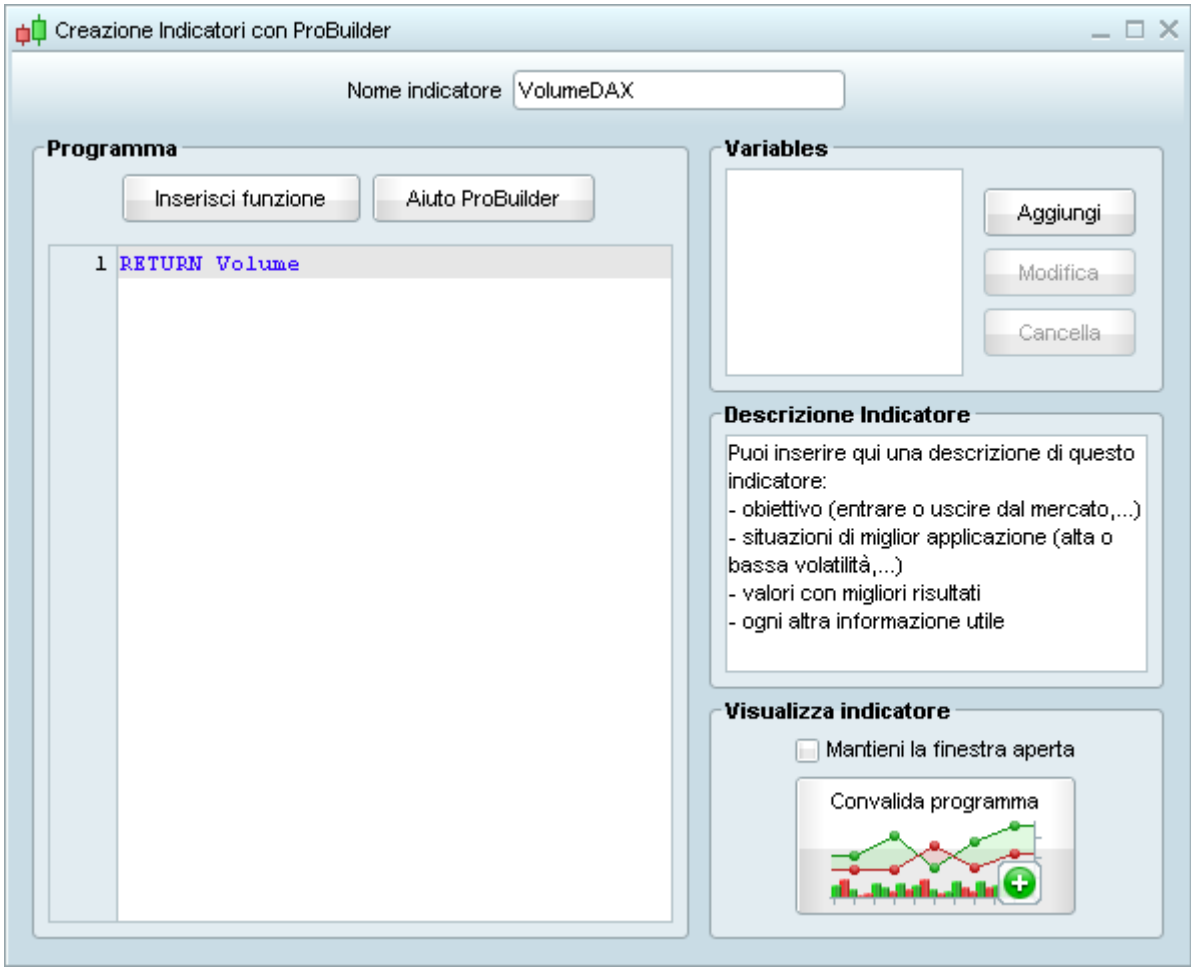
Prendiamo per esempio l'elemento caratteristico degli indicatori ProBuilder, cioé la funzione "**RETURN**" (disponibile nella sezione "Parole Chiave", come da immagine).



Selezionate la parola **RETURN** e cliccate su "Aggiungi": il comando sarà aggiunto alla zona di programmazione.

RETURN vi permette di visualizzare il risultato come indicatore

Supponiamo ora di voler creare un indicatore che illustri il Volume. Poiché abbiamo già inserito il comando **RETURN**, sarà sufficiente cliccare nuovamente su "Inserisci Funzione", nella lista selezionare "Costanti" e poi dal lato destro cliccare (scorrendo in basso nella lista) su **Volume**. Cliccate infine su "Aggiungi"



Prima di cliccare su "Convalida Programma", precisate il nome da attribuire all'indicatore personalizzato che abbiamo appena creato : in questo caso l'abbiamo chiamato "VolumeDAX". Selezionate dunque "Convalida Programma", e vedrete l'indicatore apparire al di sotto del vostro grafico del prezzo.

Scorciatoie di tastiera nella finestra di programmazione

La finestra di programmazione ha un serie di funzionalità a cui é possibile accedere dalle scorciatoie di tastiera a partire della versione 10 di ProRealTime:

- Seleziona tutto (Ctrl + A) : Seleziona tutto il testo nella finestra programmazione
- Copia (Ctrl + C) : Copia il testo selezionato
- Incolla (Ctrl + X) : Incolla il testo copiato
- Annulla (Ctrl + Z) : Annulla l'ultima azione nella finestra programmazione
- Ripristina (Ctrl + Y) : Ripristina l'ultima azione nella finestra programmazione
- Trova / Sostituisci (Ctrl + F) : Trova un testo nella finestra programmazione / sostituisci un testo nella finestra programmazione (questa funzionalità é minuscola maiuscola)
- Commenta / Decommenta (Ctrl + R) : Commenta il codice selezionato / Decommenta il codice selezionato (il codice commentato sarà preceduto da "//" o "REM" e di colore grigio. Sarà preso in considerazione quando il codice é eseguito).

Per gli utenti Mac, la stessa scorciatoia di tastiera puo' essere accessibile con il tasto "Apple" al posto del tasto "Ctrl". La maggior parte di queste funzionalità possono essere accessibili con un clic destro nella finestra di programmazione.

Specificità di programmazione del linguaggio ProBuilder

Le specificità

Il linguaggio ProBuilder permette l'utilizzo di numerosi comandi classici e più elaborati, specifici all'analisi tecnica, che vi daranno la possibilità di programmare degli indicatori dai più semplici ai più sofisticati.

I principi chiave del linguaggio ProBuilder sono :

- Non é necessario dichiarare le variabili
- Non é necessario definire il tipo di variabile
- Non c'é differenza tra minuscole e maiuscole (esiste pero' una particolarità che vedremo in seguito)
- Si utilizza lo stesso segno per l'attribuzione e l'uguaglianza matematica

Che cosa significa ?

- Dichiarare una variabile X, significa indicarne l'esistenza. In ProBuilder, potete direttamente utilizzare X senza doverne definire precedentemente l'esistenza.

  Facciamo un esempio scrivendo :

  Con dichiarazione : Abbiamo la variabile X, attribuiamo a X il valore 5

  Senza dichiarazione : Attribuiamo a X il valore 5 (quindi implicitamente, X esiste e vale 5)

  in ProBuilder basta scrivere : X=5

- **Definire il tipo di variabile**, significa definire la natura della variabile: si tratta di un intero naturale (ex: 3; 8; 21; 643; …), un intero relativo (ex: 3; 632; -37; …), un decimale (ex: 1.76453534535…), un booleano (VERO, FALSO),… ?

- In ProBuilder, é possibile scrivere i comandi indifferentemente con maiuscole e minuscole. Per esempio, l'insieme dei comandi IF / THEN / ELSE / ENDIF potrà essere scritto come iF / tHeN / ELse / endIf (e in tutte le altre varianti !)

  Eccezione: quando decidete di utilizzare una variabile, sarà necessario rispettare l'ortografia del nome definito. Se per esempio avete scritto "vARiaBILe", e desiderate richiamare questa variabile nel programma, dovrete rispettarne l'ortografia.

- Attribuire un valore ad una variabile. Per comprendere questo principio, é necessario considerare una variabile come una scatola vuota che aspetta di essere riempita. Lo schema sotto illustrato vi illustra il principio applicato al Volume:

Come potete notare, la lettura viene effettuata da destra a sinistra : il volume é attribuito a X.

Ora, per scrivere in codice ProBuilder, basta semplicemente sostituire la freccia ocn il segno =

X = Volume

Il simbolo = é utilizzato:

- Per l'attribuzione di una variabile (come nell'esempio precedente)
- Come operatore matematico binario (1+ 1= 2 é equivalente a 2 = 1 + 1).

Le costanti finanziarie ProBuilder

Prima di cominciare a programmare i vostri indicatori personali, é necessario passare in rssegna gli elementi a partire dai quali potrete costruire il vostro codice, quali i prezzi di apertura, di chiusura, il volume etc...

Si tratta dei "fondamentali" dell'analisi tecnica e della base di conoscenza essenziale per codificare gli indicatori. Potrete combinarli al fine di mettere in luce determinati aspetti dei mercati finanziari. Possiamo raggrupparli in 5 categorie:

Le constanti di prezzo e di volume che si adattano al timeframe del grafico

Si tratta delle costanti classiche più utilizzate. Riportano per defaut il valore della barra in corso (non importa l'unità di tempo) e sono presentati nel modo seguente :

- **Open :** la quotazione d'apertura della barra corrente
- **High :** il massimo della barra corrente
- **Low :** il minimo della barra corrente
- **Close :** la quotazione di chiusura di ogni barra
- **Volume :** il numero di titoli scambiati sulla barra corrente

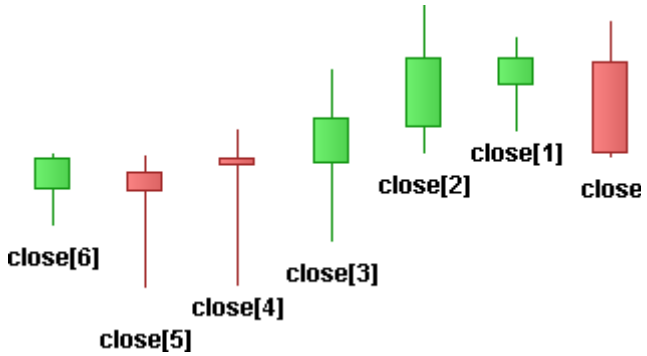
Esempio : Range della barra corrente

```
a = High
b = Low
MyRange = a - b
RETURN MyRange
```

Per richiamare i valori delle barre precedenti, é sufficiente aggiungere una parentesi quadra relativa contenente un numero relativo alla barra da considerare (a contare dalla barra in corso).

Prendiamo l'esempio della costante del prezzo di chiusura. Il richiamo alla barra del passato viene effettuato nel modo seguente :

Quotazione di chiusura della barra corrente:	*Close*
Quotazione di chiusura della barra precedente alla barra corrente:	*Close[1]*
Quotazione di chiusura della ennesima barra precedente alla barra corrente:	*Close [n]*



Questa regola vale per qualsiasi costante. Per esempio, il prezzo di apertura della seconda barra precente alla barra in corso sarà chiamato da: Open[2].

Il valore che sarà rinviato, dipenderà dal timeframe del grafico.

Le constanti giornaliere di prezzo

Contrariamente constanti di prezzo e di volume che si adattano al timeframe del grafico, le constanti giornaliere di prezzo si riferiscono ai valori della giornata, indipendentemente dal periodo illustrato sul grafico.

Una seconda differenza da constatare, consiste nel fatto che le costanti giornaliere utilizzano le parentesi per ottenere i valori delle barre del passato

- **DOpen(n) :** apertura della nesima giornata precedente al giorno in corso
- **DHigh(n) :** massimo della nesima giornata precedente al giorno in corso
- **DLow(n) :** minimo della nesima giornata precedente al giorno in corso
- **DClose(n) :** chiusura della nesima giornata precedente al giorno in corso

Attenzione: se il valore tra parentesi é uguale a 0, recuperiamo il valore del giorno in corso (come illustrato nell'esempio seguente).

Tutti i day-traders conoscono l'importanza particolare della quotazione di chiusura della vigilia (quella su cui tutti gli amatori rifletteranno durante la notte) e della quotazione d'apertura del giorno, risultante dall'emozione estrema di questi stessi amatori.

I più alti e più bassi danno anche delle indicazioni interessanti sul probabile range del giorno.

Esempio : Range giornaliero

```
a = DHigh(0)
b = DLow(0)
MyRange = a - b
RETURN MyRange
```

Per le costanti che si adattano alle unità di tempo, utilizziamo delle parentesi quadre, per le costanti giornaliere utilizziamo delle parentesi tonde.

Close [3]

DClose(3)

Le costanti temporali

Il tempo é una componente a volte non sufficientemente presa in considerazione dall'analisi tecnica.

I traders ne conoscono pero' l'importanza di determinati momenti della giornata, o di determinate date dell'anno. E' dunque possibile limitare l'analisi del proprio indicatore a dei momenti temporali specifici, utilizzando le seguenti costanti:

- **Date :** Date codificata come YYYYMMJJ, indicante la data di chiusura di ogni barra.

Le costanti temporali sono considerate da ProBuilder come degli interi. La costante Date, per esempio, deve essere presentata come un unico intero composto da 8 cifre.

Proviamo a scrivere il programma:

```
RETURN Date
```

Supponiamo che oggi sia il 4 luglio 2008, dunque 20080704, l'indicatore presenterà i numeri tutti uniti, rinviando il risultato "venti milioni ottantamila e settecento quattro".

La data in ProBuilder si basa dunque sulla scala seguente :

- Un millennio vale 10 000 000 unità di data
- Un secolo vale 1 000 000 unità di data
- Une decennio vale 100 000 unità di data
- Un anno vale 10 000 unità di data
- Dieci mesi valgono 1 000 unità di data (non superare i 12 mesi)
- Un mese vale 100 unità di data (non superare i 12 mesi)
- Dieci giorni valgono 10 unità di data (non superare i 31 giorni)
- Un giorno vale 1 unités de date (non superare i 31 giorni)

- **Time :** Ora codificata come HHMMSS indicante l'ora di chiusura di ogni barra

Facciamo un esempio :

```
RETURN Time
```

Otteniamo una curva che collega tutte le ore di chiusura di ogni barra :

E' possibile combinare in uno stesso indicatore **Time** e **Date** per restringere il risultato di un calcolo ad un momento specifico. Nell'esempio seguente, proviamo a limitare il nostro indicatore al 1° Ottobre 2008, alle 9h00 e 1 sec.

```
a = (Date = 20081001)
b = (Time = 090001)
RETURN (a AND b)
```

Lo stesso ragionamento precedente si applica alla costante Time, salvo che questa volta:

- Dieci ore valgono 100 000 unità di tempo (non superare le 23 ore)
- Una ora vale 10 000 unità di tempo (non superare le 23 ore)
- 10 minuti vale 1 000 unità di tempo (non superare i 59 minuti)
- Un minuto vale 100 unità di tempo (non superare i 59 minuti)
- 10 secondi valgono 10 unità di tempo (non superare i 59 secondi)
- 1 secondo vale 1 unité de temps (non superare i 59 secondi)

Allo stesso modo si comportano le costanti seguenti :

- **Minute :** Minuto della chiusura della barra corrente (tra 0 e 59)
- **Hour :** Ora della chiusura della barra corrente (tra 0 e 23)
- **Day :** Giorno della chiusura della barra corrente (tra 1 e 28 o 29 o 30 o 31)
- **Month :** Mese della chiusura della barra corrente (tra 1 e 12)
- **Year :** Anno della chiusura della barra corrente
- **DayOfWeek :** Giorno della settimana della chiusura della barra corrente (non tratta i giorni del weekend) : 1=lunedi, 2=martedi, 3=mercoledi, 4=giovedi, 5=venerdi

Esempio di impiego delle istruzioni

```
a = (Hour > 170000)
b = (Day = 30)
RETURN (a AND b)
```

- **CurrentHour :** Ora in corso (del mercato)
- **CurrentMinute :** Minuto in corso (del mercato)
- **CurrentMonth :** Mese in corso (del mercato)
- **CurrentSecond :** Secondo in corso (del mercato)
- **CurrentTime :** OraMinutoSecondo in corso (del mercato)
- **CurrentYear :** Anno in corso (del mercato)
- **CurrentDayOfWeek :** Giorno della settimana in corso, secondo il uso orario del mercato

La differenza tra le costanti in Current sopra proposte e senza current viste in precedenza é appunto la caratteristica dell'attualità.

L'immagine seguente illustra la differenza applicata alle costanti **CurrentTime** e **Time**. Per semplificare, le costanti in Current fanno astrazione dall'asse orizzontale del tempo. Bisognerà dunque considerare solo i valori illustrati nel riquadro bianco sull'asse verticale.

Time indica l'ora di chiusura di ogni barra

CurrentTime indica l'ora del mercato (al fuso orario di mercato)

Se desiderate impostare un indicatore utilizzando un contatore (di numero di giorni barre etc…) le costanti **Days**, **BarIndex** et **IntradayBarIndex** sono a vostra disposizione.

- **Days :** Contatore di giorni a partire dal 1900

Questa costante é utile nel momento in cui desideriamo conoscere il numero di giorni intercorsi, in particolare se lavoriamo con viste quantitative, come (x)Tick o (x)Volumi.

L'esempio seguente permetterà la visualizzazione del passaggio da una giornata all'altra di quotazione, per le viste sopra citate.

```
RETURN Days
```

- **BarIndex :** Contatore di barre dalla prima dello storico visualizzato

Il contatore parte da sinistra a destra e conta ogni barradalla barra. La prima barra caricata a sinistra é considerata barra numero 0. Il più delle volte, **BarIndex** viene utilizzato insieme all'istruzione **IF**, presentata più avanti nel manuale.

- **IntradayBarIndex :** Contatore di barre intraday

Il contatore illustra il numero di barre a partire dall'inizio della giornata e rimette il valore a zero ad ogni nuovo inizio di giornata. Come per **BaxIndex**, la prima barra a sinistra é considerata come barra 0, contrariamente al funzionamento delle altre costanti.

Compariamo dunque le due costanti **creando due indicatori separati:**

```
RETURN BarIndex
```

e

```
RETURN IntradayBarIndex
```

Notiamo come **IntradayBarIndex** viene rimesso a 0 ad ogni inizio di giornata.

Le costanti derivate dei prezzi

Queste costanti permettono di ottenere delle informazioni più complete rispetto a **Open**, **High**, **Low** e **Close**, dato che combinano queste informazioni in modo da insistere su determinati aspetti della psicologia della folla, riassunti sulla barra in corso.

- **Range :** differenza fra **High** e **Low**
- **TypicalPrice :** media fra **High**, **Low** e **Close**
- **WeightedClose :** media ponderata di **High** (peso 1) **Low** (peso 1) e **Close** (peso 2)
- **MedianPrice :** la media fra **High** e **Low**
- **TotalPrice :** la media fra **Open**, **High**, **Low** e **Close**

Il **Range** mette l'accento sulla volatilità della barra corrente, che riflette il nervosismo dei participanti.

Il **WeightedClose** insiste sull'importanza della quotazione di chiusura, quotazione di riferimento, e fisso durante tutta la durata che separa due barre consecutive (ancora più importante sulle barre giornaliere, oppure settimanali).

Le costanti **TypicalPrice** e **TotalPrice** riflettono meglio la psicologia del mercato intra-barra corrente in quanto esse prendono in conto tre e quattro livelli di quotazione raggiunti nel corso della giornata.

In un'ottica di robustezza, **MedianPrice** permette di sfruttare l'eccedenza di carattere esplicativo che offrono le mediane rispetto alle medie mobili, e si presta di più a delle modellazioni teoriche che tentano di liberarsi della psicologia della folla.

La base d'informazioni constituita da queste 5 costanti permette dunque di portare una spiegazione un pó più leggibile rispetto a (**Open**, **High**, **Low**, **Close**)

Range en % :

```
MyRange = Range
Calcolo = (MyRange / MyRange[1] - 1) * 100
RETURN Calcolo
```

La costante indefinita

La costante **Undefined** permette di imporre ad un indicatore di non attribuire alcun risultato a determinate variabili (per defaut tutte le variabili non definite hanno un valore pari a 0).

- **Undefined :** valore indefinito (equivalente a una casella vuoto o a NULL).

Potrete ritrovare qualche esempio più avanti nel manuale.

Utilizzo di indicatori pre-esistenti

Finora abbiamo osservato le possibilità offerte da ProBuilder in termini di costanti e del loro comportamento rispetto all'accesso alle barre del passato. Lo stesso comportamento si applica al funzionamento degli indicatori pre-esistenti nella piattaforma (e vedremo in seguito che i codici personalizzati che programmerete funzionano secondo lo stesso principio).

Gli indicatori ProBuilder si compongono di tre elementi la cui sintassi é :

Nome della Funzione [calcolato su n barre] (su un prezzo o un indicatore)

Il tasto "Inserisci Funzione" permetterà di ricercare una funzione ProBuilder, che apparirà con dei valori di défaut posizionati per il **periodo** e per l' argomento del **prezzo o dell'indicatore**.

Average[20](Close)

E' possibile modificare i parametri delle funzioni cosi' ottenute: per esempio, é possibile sostituire le 20 barre deinite come periodo, con un altro qualsiasi valore. Parimenti, é possibile modificare l'argomento del prezzo o la costante close per **Dopen(6)** :

Average[100](Dopen(6))

Prendiamo qualche esempio di comportamento degli indicatori pre-esistenti :

Programma che calcola la media mobile esponenziale a 20 periodi, applicato alla chiusura :

```
RETURN ExponentialAverage[20](Close)
```

Programma che calcola la media mobile ponderata a 20 periodi applicata al prezzo tipico

```
mm = WeightedAverage[20](TypicalPrice)
RETURN mm
```

Programma che calcola la media mobile lisciata di Wilder a 100 periodi, applicata al Volume

```
mm = WilderAverage[100](Volume)
RETURN mm
```

Calcolo dell'MACD (in istogramma) sul prezzo di chiusura. La linea dell'MACD si costruisce come differenza tra la media mobile esponenziale a 12 periodi meno quella a 26 periodi. Si effettua in seguito un lisciaggio con una media mobile esponenziale a 9 periodi, applicata alla differenza sopra calcolata, per ottenere la linea del Segnale.

L'MACD in istogramma di costruisce allora a partire dalla differenza tra la linea dell'MACD ed il Segnale.

```
REM Calcolo della linea MACD
LineaMACD = ExponentialAverage[12](Close) - ExponentialAverage[26](Close)
REM Calcolo della linea del Segnale MACD
LineaSegnale = ExponentialAverage[9](LineaMACD)
REM Calcolo della differenza tra la linea dell'MACD e il suo Segnale.
MACDIstogramma = LineaMACD - LineaSegnale
RETURN MACDIstogramma
```

Ottimizzazione Variabili

Quando codiamo un indicatore, introduciamo un determinato numero di costanti. L'opzione di ottimizzazione delle variabili, che troverete in alto a destra nell'interfaccia di creazione, permette di attribuire un valore di defaut alla costante indefinita e d'agire in seguito sul valore di questa costante a partire dall'interfaccia di proprietà dell'indicatore.

Il vantaggio consiste dunque nella possibilità di modicare i parametri dell'indicatore senza entrare nel codice.

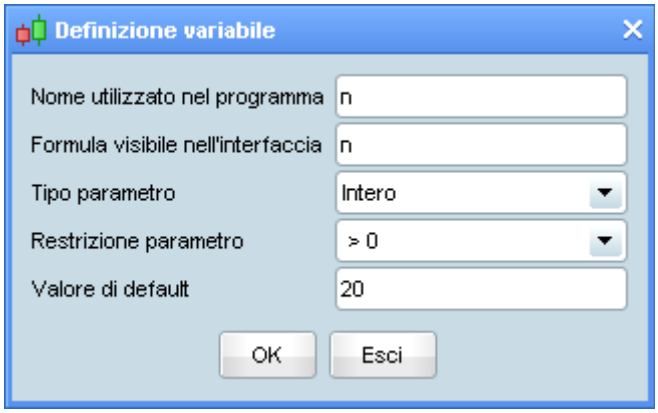
Calcoliamo per esempio una media mobile di periodo 20:

```
RETURN Average[20](Close)
```

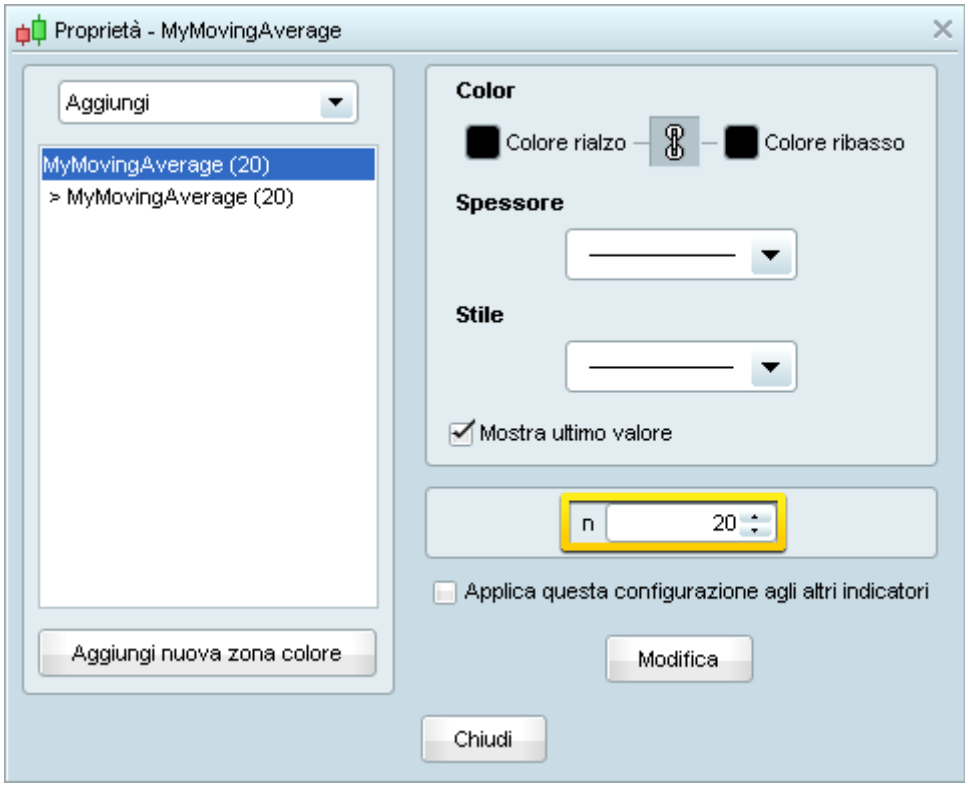
Per poter modificare il numero di periodi utilizzati per il calcolo, basterà sostituire il numero 20 con la variabile "n" :

```
RETURN Average[n](Close)
```

Cliccate in seguito su "Aggiungi" nella sezione "Ottimizzazione Variabili" e vedrete dunque apparire la finestra "Definisci Variabile". Completate come indicato:



Cliccate su "OK". Nella finestra di Proprietà dell'indicatore , potrete ora vedere una nuova sezione che permette di agire direttamente sul numero di periodi della media mobile cosi' modificata



E' naturalmente possibile effettuare l'ottimizzazione su molteplici variabili, potendo cosi' agire su svariati parametri contemporaneamente

Capitolo II : Funzioni matematiche e istruzioni ProBuilder

Strutture di controllo

Instruzioni condizionata IF

L'istruzione **IF** serve ad effettuare una scelta di azione condizionata, cioé a subordinare un risultato alla verifica di una o più condizioni definite.

La struttura di compone degli elementi **IF**, **THEN**, **ELSE**, **ELSIF**, **ENDIF**, che si combinano a seconda della complessità delle condizioni che vogliamo definire. Analizziamone insieme il modo d'uso.

Une condizione, un risultato (IF THEN ENDIF)

Abbiamo la possibilità di ricercare una condizione e di definire una azione se la condizione é verificata. Se invece la condizione non é verificata, non succede nulla (per defaut sin rinvia 0).

Nell'esempio, se l'ultimo prezzo é superiore alla MM di periodo 20, allora visualizzeremo il valore 1.

```
IF Close > Average[20](Close) THEN   Se il prezzo é > alla media mobile di 20 periodi ALLORA
   Result = 1                        Result sarà uguale a 1
ENDIF                                Fine condizione
RETURN Result                        Mostra risultato
```

RETURN deve sempre essere seguito dalla variabile contenitore (nell'esempio Result) se vogliamo che il risultato sia illustrato sul grafico

Una condizione, due risultati (IF THEN ELSE ENDIF)

Possiamo ugualmente definire un risultato nel caso la condizione non sia verificata. Riprendiamo insieme l'esempio precedente: se l'ultimo prezzo é superiore alla media mobile di periodo 20, rinvieremo 1.

Altrimenti, rinvieremo -1.

```
IF Close > Average[20](Close) THEN
   Result = 1
ELSE
   Result = -1
ENDIF
RETURN Result
```

NB: Abbiamo appena creato un indicatore binario. Per saperne di più, andate alla sezione Creare un indicatore binario o ternario : come e perché ? piu' avanti in questo manuale.

Condizioni concatenate

E' possibile creare delle condizioni concatenate a seguito di una condizione principale, cioé delle condizioni che devono essere verificate l'una dietro l'altra nell'ordine di apparizione, sulla stessa barra. Per fare cio', basta creare in sequenza gli IF, facendo pero' attenzione ad inserire tanti **ENDIF** che **IF**. Proviamo insieme:

Doppia condizione su media mobile :

```
IF (Average[12](Close) - Average[20](Close) > 0) THEN
   IF ExponentialAverage[12](Close) - ExponentialAverage[20](Close) > 0 THEN
      Result = 1
   ELSE
      Result = -1
   ENDIF
ENDIF
RETURN Result
```

Condizioni Multiple (IF THEN ELSE ELSIF ENDIF)

E' possible definire un risultato associato ad ogni condizione specifica. Avremo dunque diversi risultati associati ognuno ad una condizione specifica. L'indicatore registrerà diversi stadi : se la Condizione 1 é verificata, attiviamo l'Azione 1; altrimenti, se la Condizione 2 é verificata, attiviamo l'Azione 2...se nessuna condizione é verificata; attiviamo l'Azione n.

Sintatticamente, questa struttura utilizza le istruzioni **IF**, **THEN**, **ELSIF**, **THEN** .... **ELSE**, **ENDIF** e possiamo descriverla nel modo seguente :

```
IF (Condizione1) THEN
   (Azione1)
ELSIF (Condizione2) THEN
   (Azione2)
ELSIF (Condizione3) THEN
   (Azione3)
...
...
...
ELSE
   (Azione n)
ENDIF
```

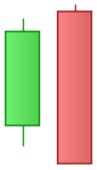
E' inoltre possibile, ma la scrittura sarebbe più pesante, sostituire **ELSIF** con **ELSE IF**. Sarà comunque necessario chiudere l'azione con altrettanti **ENDIF**.

Esempio: ricerca di un trend a rialzo e a ribasso

Questo indicatore rinvia 1 se viene identificato un trend a rialzo, -1 se a ribasso, se nulla invece é identificato, si avrà il risultato 0.

```
// Descrizione di un trend a rialzo
Condizione1 = Close[1] < Open[1]
Condizione2 = Open < Close[1]
Condizione3 = Close > Open[1]
Condizione4 = Open < Close
```

```
// Descrizione di un trend a ribasso
Condizione5 = Close[1] > Open[1]
Condizione6 = Close < Open
Condizione7 = Open > Close[1]
Condizione8 = Close < Open[1]
```



```
IF Condizione1 AND Condizione2 AND Condizione3 AND Condizione4 THEN
   a = 1
ELSIF Condizione5 AND Condizione6 AND Condizione7 AND Condizione8 THEN
   a = -1
ELSE
   a = 0
ENDIF
RETURN a
```

Esempio: Resistenza dei Pivot Demarks

```
IF DClose(1) > DOpen(1) THEN
   Phigh = DHigh(1) + (DClose(1) - DLow(1)) / 2
   Plow = (DClose(1) + DLow(1)) / 2
ELSIF DClose(1) < DOpen(1) THEN
   Phigh = (DHigh(1) + DClose(1)) / 2
   Plow = DLow(1) - (DHigh(1) - DClose(1)) / 2
ELSE
   Phigh = DClose(1) + (DHigh(1) - DLow(1)) / 2
   Plow = DClose(1) - (DHigh(1) - DLow(1)) / 2
ENDIF
RETURN Phigh , Plow
```

Esempio: BarIndex

Nel capitol 1 di questo manual, **BarIndex** é stato presentato come un contatore del numero di barre dall'inizion dello storico illustrato. **BarIndex** é spesso utilizzato in associazione con **IF**. Per esempio, volgiamo sapere se il nostro grafico contiene più o meno di 23 barre. In tal caso scriveremo:

```
IF BarIndex <= 23 THEN
   a = 0
ELSIF BarIndex > 23 THEN
   a = 1
ENDIF
RETURN a
```

L'istruzione FOR

L'istruzione **FOR** é utilizzata quando si desidera richiamare uno per uno, una serie finita di elementi. Questi ultimi potranno essere di qualsiasi tipo,se la serie é ordinata.

La struttura si compone di **FOR**, **TO**, **DOWNTO**, **DO**, **NEXT**. L'utilizzo di To o di Downto varia in funzione del richiamo in ordine crescente o descrescente degli elementi da considerare. E' importante sottolineare che **FOR** e **DO** devono contenere gli estremi dell'intervallo da passare in rassegna.

Ordine crescente (FOR, TO, DO, NEXT)

```
FOR (Variabile = Valore di inizio serie) TO Valore di fine serie DO
   (Azione)
NEXT
```

Esempio: Media Mobile Lisciata di periodo 12

Creiamo una variabile Result dove sommare una per una ogni media mobile, di periodo 11, 12, 13.

```
Result = 0
FOR Variabile = 11 TO 13 DO
   Result = Average[Variabile](Close) + Result
NEXT
REM Facciamo la media delle medie mobili dividendo Result per 3 in AverageResult.
AverageResult = Result / 3
RETURN AverageResult
```

Cerchiamo di capire cosa succede tappa per tappa :

Matematicamente, vogliamo fare la media delle medie mobili di periodo 11, 12 e 13.

Variabile assumerà dunque in successione i valori 11, 12 e 13.

Result = 0

Variabile = 11

Result assume il valore del precedente + MM11 = (0) + MM11 = (0 + MM11)

Passiamo al valore seguente

Variabile = 12

Result assume il valore del precedente Result + MM12 = (0 + M11) + MM12 = (0 + MM11 + MM12)

Passiamo al valore seguente

Variabile = 13

Result assume il valore del precedente Result + MM13 = (0 + M11 + M12) + M13 = (0 + M11 + M12 + M13)

13 é l'ultimo valore della nostra serie.

Chiudiamo "**FOR**" con l'istruzione "**NEXT**".

Visualizziamo Result

Il codice sopra illustrato prende dunque in considerazione in un primo momento il valore di inizio della seria, poi prende il valore seguente (il precedente +1) e cosi' via, fino a giungere al valore finale. Chiudiamo cosi' la serie.

Esempio: Media mobile sulle ultime 20 barre sui massimi

```
IF BarIndex < 20 THEN                Se non ci sono più di 20 periodi sullo storico
   MMhigh = Undefined                Attribuiamo a MMhigh il valore "nullo"
ELSE                                 Altrimenti
   FOR i = 0 TO 20 DO                Per i valori compresi tra 1 e 20
      SUMhigh = High[i]+SUMhigh      Sommiamo gli ultimi 20 valori dei massimi
   NEXT
ENDIF
MMhigh = SUMhigh / 20                Facciamo la media della somma e attribuiamola alla MMhigh
RETURN MMhigh                        Visualizziamo MMhigh
```

Ordine decrescente (FOR, DOWNTO, DO, NEXT)

L'ordine crescente utilizza invece le istruzioni : **FOR**, **DOWNTO**, **DO**, **NEXT**.

Si scrive nel modo seguente :

```
FOR (Variabile = Valore di fine serie) DOWNTO Valore di inizio serie DO
   (Azione)
NEXT
```

Riprendiamo l'esempio della media mobile sulle 20 ultime barre sui massimi :

Noterete che abbiamo solo invertito gli estremi dell'intervallo da considerare.

```
IF BarIndex = 0 THEN
   MMhigh = Undefined
ELSE
   FOR i = 20 DOWNTO 1 DO
      SUMhigh = High[i] + SUMhigh
   NEXT
ENDIF
MMhigh = SUMhigh / 20
RETURN Mmhigh
```

L'istruzione WHILE

WHILE serve ad applicare una condizione finché questa é valida. Noterete le similitudini con l'istruzione semplice **IF**/**THEN**/**ENDIF**.

Sintatticamente, si utilizza la struttura: **WHILE**, (**DO** facoltativo), **WEND**, che si applica nel modo seguente:

```
WHILE (Condizione) DO
    (Azione 1)
    …
    (Azione n)
WEND
```

Vediamo un esempio intuitivo :

```
Result = 0
WHILE Close > Average[20](Close) DO
   Result = 1
WEND
RETURN Result
```

Esempio: indicatore che calcoli il numero di candele a rialzo consecutive

```
Increase = (Close > Close[1])
Count = 0
WHILE Increase[Count] DO
   Count = Count + 1
WEND
RETURN Count
```

*Particolarità dell'istruzione condizionata **WHILE***

*Come per IF, il programma non tratterà **WHILE** se la condizione non é dichiarata.*

Facciamo un esempio :

```
Count = 0
WHILE i <> 11 DO
   i = i + 1
   Count = Count + 1
WEND
RETURN Count
```

WHILE in questo caso non conosce il valore d'origine di i e quindi non puo' testare se i é uguale a 10.

In questo caso non é quindi possibile definire la variabile i, attribuendo dunque per defaut il valore 0. Count non sarà allora preso in considerazione ed avremo il risultato 0.

Il codice corretto é invece:

```
i = 0
Count = 0
WHILE i <> 11 DO
   i = i + 1
   Count = Count + 1
WEND
RETURN Count
```

In questo codice, i é definito, dunque l'istruzione funzionerà correttamente ed invierà un risultato diverso da 0.

BREAK

L'istruzione **BREAK** permette di fare una uscita forzata da una istruzione continua **WHILE** o **FOR**. In entrambi i casi, é possibile combinare **BREAK** con l'istruzione **IF**.

Con WHILE

Quando cerchiamo di uscire da una istruzione **WHILE**, senza attendere di trovare una situazione che non soddisfi la condizione definita e che quindi ci faccia uscire dal cerchio, utilizziamo **BREAK** seguendo la struttura :

```
WHILE (Condizione) DO
   (Azione)
   BREAK
WEND
```

Prendiamo l'esempio di un indicatore che accumula il numero di peridi di rialzo e di ribasso consecutivi:

```
REM Indicatore di tendenza : cumula il numero di rialzi e di ribassi
REM Incrementiamo di 1 il contatore in caso di rialzo (per defaut é a zero) o decrementiamo di 1 per un ribasso
Increase = (Close - Close[1]) > 0
Indicator = 0
i = 0
WHILE Increase[i] DO
   Indicator = Indicator + 1
   i = i + 1
   BREAK
WEND
RETURN Indicator
```

Nel codice illustrato, se non avessimo utilizzato **BREAK**, il programma avrebbe continuato a testare le varie possibilità e il risultato sarebbe stato un indicatore di tendenza che avrebbe accumulato il numero di rialzi consecutivi.

Con FOR

Quando cerchiamo di uscire da una istruzione **FOR**, senza arrivare all'ultimo valore (o al primo valore della serie), utilizziamo **BREAK** seguendo la struttura :

```
FOR (Variabile = Valore di fine serie) TO Valore di inizio serie DO
   (Azione)
   BREAK
NEXT
```

Prendiamo per esempio un indicatore che cumula il numero di rialzi consecutivi del volume nelle ultime 19 barre. L'indicatore assumerà il valore 0 in caso di ribasso.

```
Indicator = 0
FOR i = 0 TO 19 DO
   IF (Volume[i] > Volume[i + 1]) THEN
      Indicator = Indicator + 1
   ELSE
      BREAK
   ENDIF
NEXT
RETURN Indicator
```

In questo caso, se non avessimo utilizzato l'istruzione **BREAK**, il programma avrebbe continuato fino a 19 (ultimo elemento della serie), anche se la condizione del volume non fosse stata valida.

Con **BREAK**, invece, non appena la condizione non é più valida, il risultato ritorna a 0.

CONTINUE

L'istruzione **CONTINUE** permette di riportare la lettura delle istruzioni **WHILE** e **FOR** all'inizio della lettura. E' spesso utilizzato in combinazione con **BREAK**, per poter dare l'ordine di uscire (con **BREAK**) o di continuare il conto (con **CONTINUE**).

Con WHILE

Creiamo un programma che accumula il numero di candele che presentano una chiusura superiore ed una apertura inferiore alle precedenti. Se la condizione non é verificata, il contatore ritornerà a zero.

```
Increase = Close > Close[1]
Count = 0
WHILE Open < Open[1] DO
   IF Increase[Count] THEN
      Count = Count + 1
      CONTINUE
   ENDIF
BREAK
WEND
RETURN Count
```

Grazie a **CONTINUE**, quando la condizione dell'**IF** é verificata, non usciamo dall'istruzione **WHILE**, il che permette di cumulare il numero di figure che verificano la condizione. Invece, senza l'istruzione **CONTINUE**, il programma uscirebbe dall'istruzione, che la condizione dell'**IF** sia verificata o meno.

Non potremmo dunque cumulare il numero di volte in cui le figure sono apparse ed il risultato sarebbe puramente binario (1,0).

Con FOR

Creiamo un programma che cumuli il numero di candele che hanno una chiusura superiore alla precedente. Se la condizione non é verificata, il contatore ritorna a 0.

```
Increase = Close > Close[1]
Count = 0
FOR i = 1 TO BarIndex DO
   IF Increase[Count] THEN
      Count = Count + 1
      CONTINUE
   ENDIF
BREAK
NEXT
RETURN Count
```

FOR permette di testare la condizione su tutto lo storico disponibile. Grazie a **CONTINUE**, quando la condizione dell'**IF** é verificata, non usciamo dall'istruzione **FOR** e non continuiamo dunque ad applicare la formula sul valore seguente dell'i. Cio" permette di cumulare il numero di figure che verificano la condizione.

Senza l'istruzione **CONTINUE**, il programma uscirebbe, che la condizione dell'**IF** sia verificata o meno. Come nel caso precedente, non potremmo cumulare il numero di volte in cui le figure sono apparse ed il risultato sarebbe puramente binario (1,0).

ONCE

L'istruzione **ONCE** serve a dichiarare "**UNA SOLA VOLTA**" una variabile.

Vi ricorderete che in generale, prima di rinviare il risultato, il linguaggio ProBuilder legge il codice per un numero di volte pari al numero di barre presenti sul grafico. **ONCE** invece :

- E' trattato dal programma una sola ed unica volta, rilettura inclusa.
- In caso di rilettura del codice da parte del linguaggio, essa conserverà i valori calcolati durante la lettura precedente.

Per meglio comprendere l'utilizzo della funzione ONCE, é necessario considerare come il linguaggio legge normalmente un qualsiasi codice.

Ritroviamo di seguito due programmi, che rinviano rispettivamente 0 e 15 e la cui sola differenza consiste nell'aggiunta di **ONCE**:

Programma 1

```
1  Count = 0
2  i = 0
3  IF i <= 5 THEN
4     Count = Count + i
5     i = i + 1
6  ENDIF
7  RETURN Count
```

Programma 2

```
1  ONCE Count = 0
2  ONCE i = 0
3  IF i <= 5 THEN
4     Count = Count + i
5     i = i + 1
6  ENDIF
7  RETURN Count
```

Vediamo come il linguaggio ha letto il codice

Programma 1 :

Il linguaggio leggerà L1 (Count = 0; i = 0), poi L2, L3, L4, L5 e L6 (Count = 0; i = 1), ritorna a L1 e rileggerà il tutto esattamente nello stesso modo. Con **RETURN**, il linguaggio opera una uscita dal circolo dopo aver letto n volte il circolo stesso. Il risultato é dunque 0, come dopo la prima lettura del circolo.

Programma 2 :

Il linguaggio leggerà L1 (Count = 0; i = 0), poi L2, L3, L4, L5, L6 (Count = 0; i = 1) ; arrivato a "**RETURN**", ricomincierà il circolo a partire da L3 (**le linee con ONCE sono trattate una sola volta**), L4, L5, L6 (Count = 1; i = 2), poi ricomincia nuovamente con (Count = 3; i = 3) e cosi' via fino a (Count = 15; i = 6). Arrivati a questo risultato, l'istruzione in **IF** non é più trattata poiché la condizione non é più valida; non resterà dunque al programma più che leggere la linea L7, che invierà il risultato 15.

Funzioni Matematiche

Funzioni unarie e binarie

Interessiamoci ora alle funzioni matematiche. Ritroverete qui illustrate le principali funzioni. Notiamo che a e b sono degli esempi di argomenti decimali. Possono essere sostituiti da una qualsiasi altra variabile.

- **MIN**(a, b) : calcola il minimo di a e di b
- **MAX**(a, b) : calcola il massimo di a e di b
- **ROUND**(a) : arrotonda all'unità a
- **ABS**(a) : calcola il valore assoluto di a
- **SGN**(a) : attribuisce il segno ad a (1 per positivo, -1 per negativo)
- **SQUARE**(a) : calcola il quadrato di a
- **SQRT**(a) : calcola la radice quadrata a
- **LOG**(a) : calcola il logaritmo di a
- **EXP**(a) : calcola l'esponenziale di a
- **COS**(a) : calcola il coseno di a
- **SIN**(a) : calcola il seno di a
- **TAN**(a) : calcola la tangente di a
- **ATAN**(a) : calcola l'arco-tangente di a

Proviamo per esempio a creare in un codice la legge matematica normale, interessante poiché utilizza la radice quadrata e l'esponenziale:

```
REM Legge Normale applicata con x = 10, ScartoTipo = 6 e Speranza = 8
REM Definiamo in variabile ottimizzata :
ScartoTipo = 6
Speranza = 8
x = 10
Indicator= EXP((1/2)*(SQUARE(x-Speranza)/ScartoTipo))/(ScartoTipo*SQRT(2/3.14))
RETURN Indicator
```

Operatori matematici comuni

- **a < b :** a é strettamente inferiore a b
- **a <= b o a =< b :** a é inferiore o uguale a b
- **a > b :** a é strettamente superiore a b
- **a >= b o a => b :** a é superiore o uguale a b
- **a = b :** a é uguale a b (o riceve il valore b)
- **a <> b :** a é diverso da b

Funzioni di comparazione grafica

- **a CROSSES OVER b :** a incrocia b a rialzo
- **a CROSSES UNDER b :** a incrocia b a ribasso

Funzioni di somma

- **cumsum :** calcola la somma di tutte le barre del grafico

La sintassi corretta di **cumsum** é :

```
cumsum (prezzo o indicatore)
```

- **summation :** calcola la somma su un numero di barre da definire

La somma viene effettuata a partire dalla barra più recente (da destra a sinistra).

La sintassi corretta é :

```
summation[N° barre](prezzo o indicatore)
```

Funzioni Statistiche

La sintassi di queste funzioni é la stessa degli indicatori o della funzione Summation, cioé:

```
lowest[N° barre](prezzo o indicatore)
```

- **lowest :** riporta il valore minimo sul periodo definito
- **highest :** riporta il valore massimo sul periodo definito
- **STD :** riporta lo scarto tipo di un valore sul periodo definito
- **STE :** riporta l'errore di scarto di un valore sul periodo definito

Operatori logici

ProBuilder dispone di 4 operatori logici :

- **NOT**(a) : NO logico
- a **OR** b : O logico
- a **AND** b : E logico
- a **XOR** b : O esclusivo

Calcolo dell'indicatore di tendenza: On Balance Volume (OBV) :

```
IF NOT((Close > Close[1]) OR (Close = Close[1])) THEN
   MyOBV = MyOBV - Volume
ELSE
   MyOBV = MyOBV + Volume
ENDIF
RETURN MyOBV
```

Istruzioni ProBuilder

- **RETURN :** rinvia il risultato
- **CustomClose :** rinvia un valore di prezzo parametrabile; per default riporta "Close"
- **CALL :** richiama una funzione precedentemente creata dall'utilizzatore
- **AS :** attribuisce un nome ai diversi risultati illustrati
- **COLOURED :** colora i tracciati in modo personalizzato

RETURN

Abbiamo già potuto vedere nel primo capitolo l'importanza dell'istruzione **RETURN**. **RETURN** presenta delle proprietà particolari che é necessario conoscere per evitare certi errori di programmazione. Esso dunque si utilizza:

- Una sola volta
- All'ultima linea del codice
- Opzionalmente insieme ad altre funzioni come AS e COLOURED
- Per visualizzare più risultati, scriviamo RETURN seguito dai risultati che desideriamo, separati da una virgola (esempio: RETURN a ,b).

REM o //

REM permette di inserire nel codice delle annotazioni, che potranno servire come commento da analizzare in seguito o come promemoria. Le annotazioni saranno cosi' ignorate dal programma. Facciamo un esempio di inserimento commenti:

```
REM questo programma rinvia la media mobile aritmentica di periodo 20 sul prezzo di chiusura.
RETURN Average[20](Close)
```

Non utilizzate caratteri speciali (esempio : é,ù,ç,ê…) in ProBuilder, nemmeno nella sezione **REM**

CustomClose

CustomClose é una costante che permette di richiamare le costanti **Close**, **Open**, **High**, **Low** ed altri valori, che possono essere selezionati nella finestra di proprietà dell'indicatore.

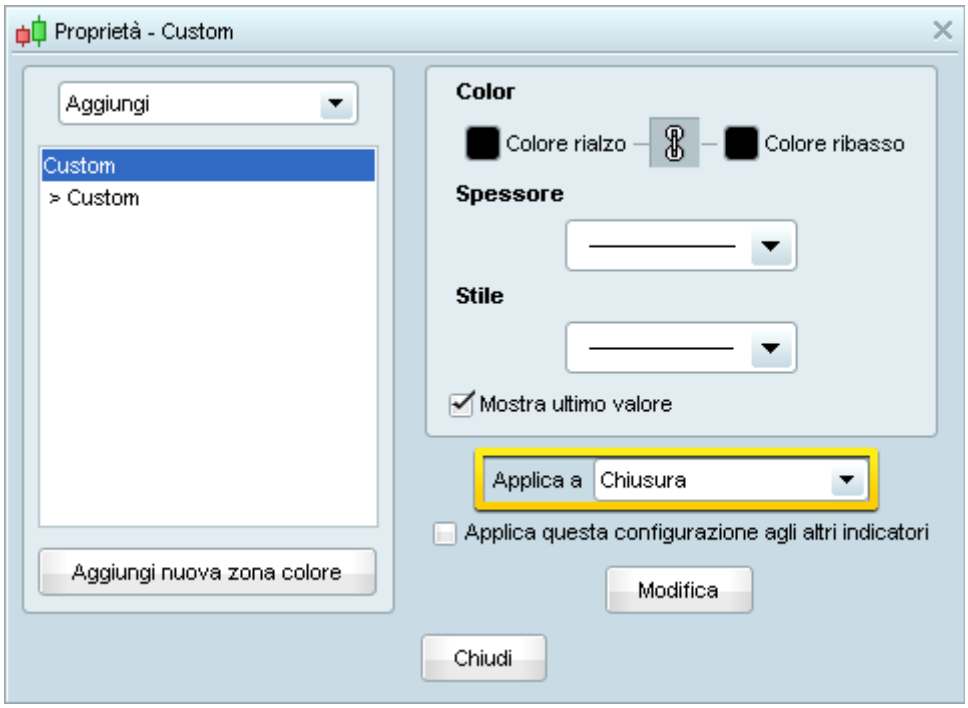
La sintassi é la stessa delle costanti di prezo che si adattano alla vista del grafico.

```
CustomClose[n]
```

Esempio:

```
RETURN CustomClose[2]
```

Cliccando sulla chiave inglese all'angolo superiore sinistro del grafico, vedrete che é possibile configurare i prezzi presi in considerazione per il calcolo (in rosso nello schema sovrastante).



CALL

CALL permette di richiamare un indicatore personalizzato già presente sulla piattaforma.

Il modo più rapido per inserire CALL consiste nel selezionare direttamente l'indicatore in questione a partire dal menu "Indicatori Utente" che si trova in "Inserisci Funzione". Immaginiamo di avere già creato un indicatore con il nome di HistoMACD (rappresentante l'MACD in istogramma).

Selezioniamo dunque l'indicatore dalla lista e clicchiamo su Aggiungi. Nella zona di programmazione apparirà :

```
myHistoMACD = CALL HistoMACD
```

Il programma a dunque rinominato il vostro precedente indicatore "HistoMACD" in "myHistoMACD".

Cio' significa che, d'ora in poi nel programma, se desideriamo utilizzare l'indicatore HistoMACD, dovremo utilizzare la sintassi "myHistoMACD".

AS

La parola chiave **AS** serve ad attribuire un nome personalizzato al risultato. Questa istruzione si utilizza con **RETURN** secondo la struttura seguente:

```
RETURN Result1 AS "Curve Name", Result2 AS "Curve Name", …
```

Il vantaggio consiste nel facilitare l'individuazione degli elementi che compongono l'indicatore creato.

Esempio:

```
a = ExponentialAverage[200](Close)
b = WeightedAverage[200](Close)
c = Average[200](Close)
RETURN a AS "Exponential Average", b AS "Weighted Average", c AS "Arithmetical Average"
```

COLOURED

COLOURED é utilizzato dopo l'istruzione **RETURN** per colorare i tracciati con un colore personalizzato, definito secondo la norma RGB (red, green, blue). Possiamo illustrarne le principali regole :

Couleur	Valori RGB (rosso, verde, blu)	Italiano
	(0, 0, 0)	nero
	(255, 255, 255)	bianco
	(255, 0, 0)	rosso
	(0, 255, 0)	verde
	(0, 0, 255)	blu
	(255, 255, 0)	giallo
	(0, 255, 255)	azzurro
	(255, 0, 255)	magenta

La sintassi di Coloured é la seguente :

```
RETURN Indicator COLOURED(Red, Green, Blue)
```

L'istruzione **AS** puo' essere associata a **COLOURED**(. , . , .) rispettando l'ordine seguente :

```
RETURN Indicator COLOURED(Red, Green, Blue) AS "Nome da attribuire"
```

Riprendiamo l'esempio precedente ed inseriamo **COLOURED** dopo "**RETURN**".

```
a = ExponentialAverage[200](Close)
b = WeightedAverage[200](Close)
c = Average[200](Close)
RETURN a COLOURED(255, 0, 0) AS "Exponential Moving Average", b COLOURED(0, 255,
0) AS "WeightedMoving Average", c COLOURED(0, 0, 255) AS "Simple Moving Average"
```

L'immagine seguente vi mostra la differenza nella visualizzazione del risultato.

Capitolo III : Applicazioni pratiche

Creare un indicatore binario o ternario : come e perché ?

Un indicatore binario o ternario é per definizione un indicatore che puo' proporre solo due o tre risultati possibili (di solito 0,1 e -1). L'utilità principale, nell'ambito borsistico, consiste nel rendere immediatamente identificabile la verifica della condizione che costituisce l'indicatore.

Utilità di un indicatore binario o ternario :

- Permettere di ritrovare le principali figure di candele giapponesi (pattern di prezzo).
- Facilitare la lettura del grafico quando si ricercano diverse condizioni contemporanee.
- Poter inserire degli allarmi a una sola condizione su un indicatore che ne incorpora diverse ➡ avrete dunque più allarmi a disposizione !
- Ritrovare delle condizioni complesse anche sullo storico
- Facilitare la realizzazione di un backtest

Gli indicatori binari o ternari sono costruiti attraverso la funzione IF. Consigliamo dunque la lettura della sezione relativa prima di continuarne con il presente capitolo.

Illustriamo la creazione di un indicatore personalizzato per trovare i pattern di prezzo :

Indicatore binario: Martello

```
// Troviamo il martello
Hammer = Close>Open AND High = Close AND (Open-Low) >= 3*(Close-Open)
IF Hammer THEN
   Result = 1
ELSE
   Result = 0
ENDIF
RETURN Result AS "Hammer"
```

Indicatore ternairio: Croce dorata e croce mortale

```
a = ExponentialAverage[10](Close)
b = ExponentialAverage[20](Close)
c = 0
// Troviamo la croce dorata
IF a CROSSES OVER b THEN
   c = 1
ENDIF
// Troviamo la croce mortale
IF a CROSSES UNDER b THEN
   c = -1
ENDIF
RETURN c
```

Attenzione : abbiamo illustrato le medie mobili esponenziali di periodo 10 e 20, applicate al prezzo di chiusura per meglio mettere in evidenza la corrispondenza dei risultati dell'indicatore.

Potrete ritrovare altri esempi di indicatori per l'individuazione dei pattern di prezzo, nella alla sezione "Capitolo IV: esercizi" più avanti in questo manuale.

Creare degli indicatori STOP : seguite le vostre posizioni in tempo reale

E' possibile creare degli indicatori che rappresentano degli STOP, cioé dei livelli di uscita potenziale, definiti secondo dei parametri personalizzati.

Attraverso il modulo ProBackTest, che é illustrato in un manual indipendente, é già possibile definire i livelli di uscita di una strategia. L'interesse pero' della programmazione di un indicatore che segue il corso del prezzo, risiede nel fatto che:

- Permette di visualizzare la linea aggiornata in tempo reale direttamente sul grafico del prezzo
- Non é necessario associare gli ordini di acquisto e di vendita associati (necessario nel ProBackTest)
- E' possibile associarvi degli allarmi in tempo reale, per essere avvertiti in tempo reale della verifica della condizione.

La programmazione degli Stop vi permetterà di impiegare gli strumenti illustrati nei capitoli precedenti.

Il manuale di programmazione ProBackTest vi permetterà inoltre di trovare degli esempi di stop inseriti in strategie di investimento

Esistono 4 categorie di stop che esamineremo :

- **TAKE PROFIT (STOP statico di guadagno)**
- **STOP LOSS (statico)**
- **STOP d'inattività**
- **TRAILING STOP (dinamico)**

I codici proposti negli esempi che seguono, rappresentano dei suggerimenti per la costruzione degli indicatori di stop. Dovrete necessariamente personalizzarli utilizzando le istruzioni illustrate nei capitoli precedenti.

TAKE PROFIT

Un *Take-Profit* o *STOP statico di guadagno*, designa un limite superiore di uscita di posizione. Questo limite per definizione é fisso. Il trader otterrà un guadagno chiudendo la posizione aperta quando il prezzo incrocia la linea di STOP.

L'indicatore presentato qui in seguito, propone due livelli di posizione, long e short, a partire dalla data "Start".

- Se siete long (dunque in acquisto), terrete conto della curva superiore, che rappresenta il 10% di guadagno, cioé almeno il 110% del prezzo al momento dell'acquisto.
- Se siete short (vendita allo scoperto), terrete conto della curva inferiore, che rappresenta il 10% di guadagno, cioé almeno il 90% del prezzo al momento della vendita allo scoperto.

Esempio di STOP personalizzabile

```
// Definiamo la variabile ottimizzata
// StartingTime = 100000
// Price = Prezzo di apertura della posizione (nell'esempio, data di inizio posizione definita alla 10h00)
// Se siete long, considererete la curva superiore; se short, la curva inferiore.
// AmpiezzaUp = 1.1 (rappresenta la variazione di Prezzo, usata per tracciare la posizione long)
// AmpiezzaDown = 0.9 (rappresenta la variazione di Prezzo, usata per tracciare la posizione short)
IF Time = StartingTime THEN
   StopLONG = AmpiezzaUp * Price
   StopSHORT = AmpiezzaDown * Price
ENDIF
RETURN StopLONG COLOURED(0, 0, 0) AS "TakeProfit LONG", StopSHORT COLOURED(0, 255, 0) AS "TakeProfit SHORT"
```

STOP LOSS statico

Uno STOP Loss é il contrario dello STOP Take-Profit, dunque definisce un livello di perdita massima a partire dal quale on chiuderà la posizione, nonostante sia in perdita.

E' particolarmente utile per limitare le perdite ad un montante massimo. Il limite é fisso.

L'indicatore che presentiamo di seguito indica due livelli di presa di posizione alla data "Start".

- Se siete long (dunque in acquisto), terrete conto della curva inferiore, che rappresenta il 10% di perdita, cioé almeno il 90% del prezzo al momento dell'acquisto.
- Se siete short (vendita allo scoperto), terrete conto della curva superiore, che rappresenta il 10% di perdita, cioé almeno il 110% del prezzo al momento della vendita allo scoperto.

Un esempio segue :

```
// Definiamo la variabile ottimizzata
// StartingTime = 100000
// Price = Prezzo di apertura della posizione (nell'esempio, data di inizio posizione definita alla 10h00)
// Se siete long, considererete la curva inferiore; se short, la curva inferiore.
// AmpiezzaUp = 0.9 (rappresenta la variazione di Prezzo, usata per disegnare la posizione long per Take Profit)
// AmpiezzaDown = 1.1 (rappresenta la variazione di Prezzo, usata per disegnare la posizione short per Take Profit)
IF Time = StartingTime THEN
   StopLONG = AmpiezzaUp * Price
   StopSHORT = AmpiezzaDown * Price
ENDIF
RETURN StopLONG COLOURED(0, 0, 0) AS "StopLoss LONG", StopSHORT COLOURED(0, 255, 0) AS "StopLoss SHORT"
```

STOP d'inattività

Uno STOP di inattività chiude una posizione quando l'obiettivo di profitto (in % o in punti) non é raggiunto in un periodo determinato (espresso in numero di barre).

Lo STOP che proponiamo di seguito si compone di due indicatori: un indicatore da inserire nel garfico del prezzo, per valutare l'obiettivo di profitto ed un indicatore binario che rinvia 1 se lo STOP viene attivato, altrimenti vale 0.

Indicatore 1

```
// MyVolatility = 0.01 corrisponde allo scarto relativo delle bande superiori ed inferiori
IF IntradayBarIndex = 0 THEN
   ShortTarget = (1 - MyVolatility) * Close
   LongTarget = (1 + MyVolatility) * Close
ENDIF
RETURN ShortTarget AS "ShortTarget", LongTarget AS "LongTarget"
```

Indicatore 2

```
// Variabili da ottimizzare :
REM L'acquisto si fa a prezzo di mercato
// MyVolatility = 0.01 corrisponde allo scarto relativo delle bande superiori ed inferiori del range definito.
// NumberOfBars = 20 corrisponde alla durata massima in numero di barre permessa prima di forzare la chiusura della posizione (risultato a 1)
Result = 0
Cpt = 0
IF IntradayBarIndex = 0 THEN
   ShortTarget = (1 - MyVolatility) * Close
   LongTarget = (1 + MyVolatility) * Close
ENDIF
FOR i = IntradayBarIndex DOWNTO 1 DO
   IF Close[i] >= ShortTarget AND Close[i] <= LongTarget THEN
      Cpt = Cpt + 1
   ELSE
      Cpt = 0
   ENDIF
   IF Cpt = NumberOfBars THEN
      Result = 1
   ENDIF
NEXT
RETURN Result
```

TRAILING STOP (dinamico)

Un *trailing STOP* segue in modo dinamico l'evoluzione del prezzo ed indica il momento in cui la posizione deve essere chiusa.

Vi proponiamo in seguito due tipi di TRAILING STOP, corrispondenti alle versioni dinamiche dello stop loss e del take profit.

Trailing STOP LOSS (intraday)

```
// Variabili da ottimizzare :
// StartingTime = 090000 (definiamo l'ingresso a mercato a partire dalle 9h00)
// la posizione viene presa a prezzo di mercato
// Ampiezza = 0.9 (rappresenta uno stop al 10%)
IF Time = StartingTime THEN
   IF lowest[5](Close) < 1.2 * Low THEN
      IF lowest[5](Close) >= Close THEN
         Cut = Ampiezza * lowest[5](Close)
      ELSE
         Cut = Ampiezza * lowest[20](Close)
      ENDIF
   ELSE
      Cut = Ampiezza * lowest[20](Close)
   ENDIF
ENDIF
RETURN Cut AS "Trailing Stop Loss"
```

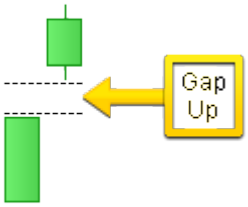
Trailing STOP Profit (da utilizzare in trading intraday)

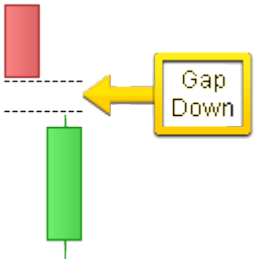
```
// Variabili da ottimizzare :
// StartingTime = 090000 (definiamo l'ingresso a mercato a partire dalle 9h00)
// la posizione viene presa a prezzo di mercato
// Ampiezza = 1.10 (rappresenta uno stop al 110%)
IF Time = StartingTime THEN
   StartingPrice = Close
ENDIF
Price = StartingPrice - AverageTrueRange[10]
TrailingStop = Ampiezza * highest[15](Price)
RETURN TrailingStop COLOURED (255, 0, 0) AS "Trailing take profit"
```

Capitolo IV : esercizi

Pattern di prezzo

- **GAP UP o DOWN**





Il colore della candela non ha importanza.

Definiamo l'ampiezza come variabile ottimizzata 0.001

Un gap viene definito da due condizioni:

- l'apertura del giorno é strettamente superiore alla chiusura del giorno precedente OPPURE ou strettamente inferiore alla chiusura del giorno precedente
- [(valore assoluto dell'apertura del giorno - chiusura del giorno precedente) / chiusura del giorno precedente] strettamente superiore all'ampiezza

```
// Variabili da ottimizzare :
// Ampiezza = 0.001
// prima condizione di esistenza del gap
IF Open > Close[1] OR Open < Close[1] THEN
   // seconda condizione di esistenza del gap
   IF ABS((Open - Close[1]) / Close[1]) > Ampiezza THEN
      // Ricerca dei gap
      Detector = SGN(Open - Close[1])
   ELSE
      Detector = 0
   ENDIF
ELSE
   Detector = 0
ENDIF
// Risultato
RETURN Detector AS "Gap detection"
```

- **Doji (versione larga)**

Definiamo il doji come un range strettamente superiore a 5 volte il valore assoluto di (Open-Close)

```
Doji = Range > ABS(Open - Close) * 5
RETURN Doji AS "Doji"
```

- **Doji (versione stretta)**

Definiamo il doji come Close = Open

```
Doji = (Open = Close)
RETURN Doji AS "Doji"
```

Indicatori

- **BODY MOMENTUM**

Il Body Momentum é definito matematicamente come :

BodyMomentum = 100 * BodyUp / (BodyUp + BodyDown)

BodyUp (risp. BodyDown) é un contatore del numero di barre che riportano una chiusura maggiore (risp.inferiore) dell'apertura, su un periodo definito (per esempio periodo = 14)

```
Periods = 14
b = Close - Open
IF BarIndex > Periods THEN
   Bup = 0
   Bdn = 0
   FOR i = 1 TO Periods
      IF b[i] > 0 THEN
         Bup = Bup + 1
      ELSIF b[i] < 0 THEN
         Bdn = Bdn + 1
      ENDIF
   NEXT
   BM = (Bup / (Bup + Bdn)) * 100
ELSE
   BM = Undefined
ENDIF
RETURN BM AS "Body Momentum"
```

ONDE DI ELLIOT

L'oscillatoure Elliot rappresenta la differenza di medie mobili.

a: periodi di media mobile corta (5 per default)

b: periodi di media mobile lunga (35 per default)

Questo oscillatore permette di distinguere tra onda 3 e onda 5 usando la teoria delle onde di Elliot.

La media mobile corta rappresenta l'azione dei prezzi, mentre la media mobile lunga rappresenta la tendenza di fondo

Quando i prezzi formano una onda 3, aumenteranno fortemente e di conseguenza l'oscillatore aumenterà sensibilmente.

In una onda 5, i prezzi aumentano più lentamente e l'oscillatore riporterà dunque un valore meno elevato.

```
RETURN Average[5](MedianPrice) - Average[35](MedianPrice) AS "Elliot Wave Oscillator"
```

Williams %R

Il suo funzionamento é simile allo Stocastico. Per tracciarlo, é necessario tracciare innanzitutto due curve. Il %R é definito allora come (Close – LowestL) / (HighestH – LowestL) * 100

```
HighestH = highest[14](High)
LowestL = lowest[14](Low)
MyWilliams = (Close - LowestL) / (HighestH - LowestL) * 100
RETURN MyWilliams AS "Williams %R"
```

Bande di Bollinger

Le bande sono composte da una media mobile a 20 periodi applicata alla chiusura. La media mobile é moltiplicata per due volte lo scarto tipo

```
a = Average[20](Close)
// Definiamo lo scarto tipo
StdDeviation = STD[20](Close)
Bsup = a + 2 * StdDeviation
Binf = a - 2 * StdDeviation
RETURN a AS "Media", Bsup AS "Bollinger Up", Binf AS "Bollinger Down"
```

Glossario

A

Codice	Implementazione	Descrizione
ABS	ABS(a)	Funzione matematica "Valore assoluto"
AccumDistr	AccumDistr(price)	Accumulazione Distribuzione classica
ADX	ADX[N]	Indicatore Average Directional Index
ADXR	ADXR[N]	Indicatore Average Directional Index Rate
AND	a AND b	Operatore logico ET
AroonDown	AroonDown[P]	Aroon Down
AroonUp	AroonUp[P]	Aroon Up
ATAN	ATAN(a)	Funzione matematica Arco Tangente
AS	RETURN Result AS "ResultName"	Istruzione per attribuire un nome personalizzato
Average	Average[N](price)	Media Mobile Aritmetica
AverageTrueRange	AverageTrueRange[N](price)	Media mobile lisciata da Wilder

B

Codice	Implementazione	Descrizione
BarIndex	BarIndex	Numero di barre dall'inizio dei dati caricati (in un grafico nel caso di un indicatore ProBuilder o per un sistema di trading nel caso di ProBacktest o ProInvest)
BollingerBandWidth	BollingerBandWidth[N](price)	Banda passante di Bollinger
BollingerDown	BollingerDown[N](price)	Supporto della banda di Bollinger
BollingerUp	BollingerUp[N](price)	Resistenza della banda di Bollinger
BREAK	(FOR...DO...BREAK...NEXT) o (WHILE...DO...BREAK...WEND)	Istruzione per forzare l'uscita da un circolo FOR o WHILE

C

Codice	Implementazione	Descrizione
CALL	myResult = CALL myFunction	Permette di richiamare un altro indicatore
CCI	CCI[N](price) o CCI[N] applicato per defaut al TypicalPrice	Commodity Channel Index
ChaikinOsc	ChaikinOsc[Ch1, Ch2](price)	Oscillatore di Chaikin
Chandle	Chandle[N](price)	Chande Momentum Oscillator
ChandeKrollStopUp	ChandeKrollStopUp[Pp, Qq, X]	Stop di protezione di Chande e Kroll in acquisto
ChandeKrollStopDown	ChandeKrollStopDown[Pp, Qq, X]	Stop di protezione di Chande e Kroll in vendita
Close	Close[N]	Designa il prezzo di chiusura oppure l'ultimo prezzo registrato
COLOURED	RETURN Result COLOURED(R,G,B)	Permette di personalizzare il colore di una curva, secondo la convenzione RGB
COS	COS(a)	Coseno
CROSSES OVER	a CROSSES OVER b	Operatore boleano di incrocio a rialzo
CROSSES UNDER	a CROSSES UNDER b	Operatore boleano di incrocio a ribasso
cumsum	cumsum(price)	Somma dall'inizio dello storico
CurrentDayOfWeek	CurrentDayOfWeek	Designa il giorno attuale
CurrentHour	CurrentHour	Designa l'ora attuale
CurrentMinute	CurrentMinute	Designa il minuto attuale
CurrentMonth	CurrentMonth	Designa il mese attuale
CurrentSecond	CurrentSecond	Designa il secondo attuale
CurrentTime	CurrentTime	Designa l'ora minuto attuale
CurrentYear	CurrentYear	Designa l'anno attuale
CustomClose	CustomClose[N]	Costante parametrabile nella finestra proprietà
Cycle	Cycle(price)	Cycle

D

Codice	Implementazione	Descrizione
Date	Date[N]	Data di chiusura della barra corrente
Day	Day[N]	Giorno di chiusura della barra corrente
Days	Days[N]	Contatore di giorni dal 1900
DayOfWeek	DayOfWeek[N]	Giorno della settimana di chiusura della barra corrente
DClose	DClose(N)	Prezzo di chiusura giornaliero
DEMA	DEMA[N](price)	Doppia media mobile esponenziale
DHigh	DHigh(N)	Prezzo massimo giornaliero
DI	DI[N](price)	Demand Index
DIminus	DIminus[N](price)	DI-
DIplus	DIplus[N](price)	DI+
DLow	DLow(N)	Prezzo minimo di chiusura della barra giornaliera
DO	Vedere FOR e WHILE	Istruzione facoltativa per FOR e WHILE
DOpen	DOpen(N)	Prezzo di apertura della barra giornaliera
DOWNTO	Vedere FOR	Istruzione di lettura decrescente per FOR
DPO	DPO[N](price)	Detrented Price Oscillator

E

Codice	Implementazione	Descrizione
EaseOfMovement	EaseOfMovement[I]	Ease of Movement
ELSE	Vedere IF/THEN/ELSE/ENDIF	Istruzione per introdurre la condizione alternativa alla prima in IF
ELSEIF	Vedere IF/THEN/ELSE/ENDIF	Istruzione per introdurre una condizione doppia in IF
EMV	EMV[N]	Ease of Movement Value
ENDIF	Vedere IF/THEN/ELSE/ENDIF	Istruzione da utilizzare alla fine di IF
EndPointAverage	EndPointAverage[N](price)	Media mobile all'ultimo punto
EXP	EXP(a)	Funzione matematica Esponenziale
ExponentialAverage	ExponentialAverage[N](price)	Media mobile Esponenziale

F - G

Codice	Implementazione	Descrizione
FOR/TO/NEXT	FOR i = a TO b DO a NEXT	Elementi dell'istruzione POUR
ForceIndex	ForceIndex(price)	Force Index: determina il controllo del mercato (vendita o acquisto)

H

Codice	Implementazione	Descrizione
High	High[N]	Designa il prezzo massimo della barra
highest	highest[N](price)	Designa il prezzo massimo di un insieme di barre
HistoricVolatility	HistoricVolatility[N](price)	Volatilità Storica (o volatilità statistica)
Hour	Hour[N]	Ora di chiusura della barra corrente

I - J - K

Codice	Implementazione	Descrizione
IF/THEN/ENDIF	IF a THEN b ENDIF	Istruzioni condizionali senza seconda condizione
IF/THEN/ELSE/ENDIF	IF a THEN b ELSE c ENDIF	Istruzioni condizionali
IntradayBarIndex	IntradayBarIndex[N]	Conta il numero di candele in un grafico giornaliero

L

Codice	Implementazione	Descrizione
LinearRegression	LinearRegression[N](price)	Retta di regressione lineare
LinearRegressionSlope	LinearRegressionSlope[N](price)	Inclinazione di regressione lineare
LOG	LOG(a)	Funzione logaritmica
Low	Low[N]	Designa il prezzo minimo della barra
lowest	lowest[N](price)	Designa il prezzo minimo di un insieme di barre

M

Codice	Implementazione	Descrizione
MACD	MACD[S,L,Si](price)	Moving Average Convergence Divergence (MACD) in istogramma
MACDline	MACDLine[S,L](price)	Linea dell'MACD
MassIndex	MassIndex[N]	Mass Index
MAX	MAX(a,b)	Funzione matematica "Massimo"
MedianPrice	MedianPrice	Media del prezzo massimo e minimo
MIN	MIN(a,b)	Funzione matematica "Minimo"
Minute	Minute	Minuto di chiusura della barra corrente
MOD	a MOD b	Funzione matematica "Resto della divisione euclidiane"
Momentum	Momentum[I]	Momentum
MoneyFlow	MoneyFlow[N](price)	MoneyFlow tra -1 e 1
MoneyFlowIndex	MoneyFlowIndex[N]	MoneyFlowIndex
Month	Month[N]	Mese di chiusura della barra corrente

N

Codice	Implementazione	Descrizione
NEXT	Vedere FOR/TO/NEXT	Istruzione per chiudere il circolo FOR
NOT	NOT a	Operatore logico NON

O

Codice	Implementazione	Descrizione
OBV	OBV(price)	On-Balance-Volume
ONCE	ONCE VariableName = VariableValue	Istruzione per indicare che l'istruzione sarà eseguita 1 volta sola
Open	Open[N]	Prezzo di apertura
OR	a OR b	Operatore logico O

P - Q

Codice	Implementazione	Descrizione
PriceOscillator	PriceOscillator[S,L](price)	Percentage Price oscillator
PositiveVolumeIndex	PriceVolumeIndex(price)	Positive Volume Index
PVT	PVT(price)	"Price Volume Trend"

R

Codice	Implementazione	Descrizione
R2	R2[N](price)	Coefficiente della radice quadrata (tasso d'errore dei prezzi della regressione lineare)
Range	Range[N]	Range (differenza tra massimo e minimo della barra corrente)
REM	REM comment	Introduce un commento (non considerato nel codice)
Repulse	Repulse[N](price)	Repulse (misura la spinta a rialzo e a ribasso di ogni candela)
RETURN	RETURN Result	Istruzione che rinvia il risultato
ROC	ROC[N](price)	Price Rate of Change
RSI	RSI[N](price)	Relative Strength Index
ROUND	ROUND(a)	Funzione matematica "Arrotondamento all'unità" (Intero)

S

Codice	Implementazione	Descrizione
SAR	SAR[At,St,Lim]	Parabolic SAR
SARatdmf	SARatdmf[At,St,Lim](price)	Parabolique SAR a prezzo limite
SIN	SIN(a)	Funzione matematica "Seno"
SGN	SGN(a)	Funzione matematica "Segno"
SMI	SMI[N,SS,DS](price)	Stochastic Momentum Index
SmoothedStochastic	SmoothedStochastic[N,K](price)	Stocastico lisciato
SQUARE	SQUARE(a)	Elevazione al quadrato
SQRT	SQRT(a)	Radice quadrata
STD	STD[N](price)	Scarto tipo
STE	STE[N](price)	Scarto Errore
Stochastic	Stochastic[N,K](price)	Linea %K dello Stocastico
summation	summation[N](price)	Somma del prezzo delle ultime N candele
Supertrend	Supertrend[STF,N]	Super Trend

T

Codice	Implementazione	Descrizione
TAN	TAN(a)	Tangente
TEMA	TEMA[N](price)	Media Mobile Esponenziale Tripla
THEN	Vedere IF/THEN/ELSE/ENDIF	Istruzione che segue l'istruzione condizionale "IF"
Time	Time[N]	Permette di richiamare l'ora, in formato OraMinutoSecondo
TimeSeriesAverage	TimeSeriesAverage[N](price)	Media Mobile delle serie temporali
TO	Vedere FOR/TO/NEXT	Istruzione "fino a" nell'istruzione FOR
Today	Today	Data del giorno corrente
TotalPrice	TotalPrice[N]	(Chiusura + Apertura + Massimo + Minimo) / 4
TR	TR(price)	True Range
TriangularAverage	TriangularAverage[N](price)	Media Mobile Triangolare
TRIX	TRIX[N](price)	Tripla Media Mobile Esponenziale
TypicalPrice	TypicalPrice[N]	Prezzo Tipico (media dei massimi, minimi e chiusura)

U

Codice	Implementazione	Descrizione
Undefined	a = Undefined	Permette di lasciare una variabile indefinita

V

Codice	Implementazione	Descrizione
Variation	Variation(price)	Differenza tra la chiusura della vigilia e la chiusura corrente, in %
Volatility	Volatility[S, L]	Volatilità di Chaikin
Volume	Volume[N]	Volume
VolumeOscillator	VolumeOscillator[S,L]	Oscillatore di Volume
VolumeROC	VolumeROC[N]	Volume del Rate of Change

W

Codice	Implementazione	Descrizione
WeightedAverage	WeightedAverage[N](price)	Media Mobile Ponderata
WeightedClose	WeightedClose[N]	Media tra chiusura, massimo, minimo ponderati
WEND	Vedere WHILE/DO/WEND	Istruzione da posizionare alla fine dell'istruzione WHEN
WHILE/DO/WEND	WHILE (condition) DO (action) WEND	Istruzione WHEN
WilderAverage	WilderAverage[N](price)	Media Mobile di Wilder
Williams	Williams[N](close)	Calcula il %R di Williams
WilliamsAccumDistr	WilliamsAccumDistr(price)	Indicatore Accumulazione/Distribuzione di Williams

X

Codice	Implementazione	Descrizione
XOR	a XOR b	Operatore logico esclusivo O

Y

Codice	Implementazione	Descrizione
Year	Year[N]	Permette di richiamare un anno in particolare nel programma
Yesterday	Yesterday[N]	Permette di richiamare il giorno precedente nel programma

Z

Codice	Implementazione	Descrizione
ZigZag	ZigZag[Zr](price)	Zig-Zag della teoria delle onde di Elliot
ZigZagPoint	ZigZagPoint[Zp](price)	Zig-Zag della teoria delle onde di Elliot calcolata a Zp punti

Operatori

Codice	Descrizione	Codice	Descrizione
+	Somma	<>	Differenza
-	Sottrazione	<	Strettamente inferiore
*	Moltiplicazione	>	Strettamente superiore
/	Divisione	<=	Inferiore
=	Uguale	>=	Superiore

ProRealTime.com

Il riferimento dei software di borsa online