




Manuale di Programmazione Funzioni di base & Indicatori (ProBuilder)






ProBuilder è un linguaggio di programmazione di tipo BASIC che consente di creare indicatori tecnici personalizzati, sistemi di trading e programmi di scansione di mercato per la piattaforma ProRealTime.

Segui [ProRealTime Programming](#) su Google+ per le novità sul linguaggio di programmazione di ProRealTime.

SOMMARIO

 Presentazione del ProBuilder	1
 Capitolo 1: Le nozioni fondamentali	2
➔ Usare ProBuilder.....	2
› Supporto veloce per la creazione di indicatori.....	2
› Scorciatoie di tastiera nella finestra di programmazione.....	5
➔ Specificità di programmazione del linguaggio ProBuilder.....	6
➔ Le costanti finanziarie ProBuilder.....	7
› Le costanti di prezzo e di volume che si adattano al timeframe del grafico.....	7
› Le costanti giornaliere di prezzo.....	8
› Le costanti temporali.....	8
› Le costanti derivate dei prezzi.....	12
› La costante indefinita.....	12
➔ Utilizzo di indicatori pre-esistenti.....	12
➔ Ottimizzazione Variabili.....	13
 Capitolo II: Funzioni matematiche e istruzioni ProBuilder	15
➔ Strutture di controllo.....	15
› Istruzioni condizionata IF.....	15
• Una condizione, un risultato (IF THEN ENDIF).....	15
• Una condizione, due risultati (IF THEN ELSE ENDIF).....	15
• Condizioni concatenate.....	15
• Condizioni Multiple (IF THEN ELSE ELSIF ENDIF).....	16
› L'istruzione FOR.....	17
• Ordine crescente (FOR, TO, DO, NEXT).....	17
• Ordine decrescente (FOR, DOWNTO, DO, NEXT).....	18
› L'istruzione WHILE.....	19
› BREAK.....	20
• Con WHILE.....	20
• Con FOR.....	20
› CONTINUE.....	21
• Con WHILE.....	21
• Con FOR.....	21
› ONCE.....	22
➔ Funzioni Matematiche.....	23
› Funzioni unarie e binarie.....	23
› Operatori matematici comuni.....	23
› Funzioni di comparazione grafica.....	23
› Funzioni di somma.....	24
› Funzioni Statistiche.....	24
➔ Operatori logici.....	24

➔ Istruzioni ProBuilder.....	24
> RETURN.....	25
> REM o //.....	25
> CustomClose.....	25
> CALL.....	26
> AS.....	26
> COLOURED.....	26
➔ Istruzioni di disegno.....	28
 Capitolo III: Applicazioni pratiche	31
➔ Creare un indicatore binario o ternario: come e perché?.....	31
➔ Creare degli indicatori STOP: seguite le vostre posizioni in tempo reale.....	32
> TAKE PROFIT.....	33
> STOP LOSS statico.....	33
> STOP d'inattività.....	34
> TRAILING STOP (dinamico).....	35
 Capitolo IV: esercizi	36
➔ Pattern di prezzo.....	36
➔ Indicatori.....	37
 Glossario	39

Avvertimento: ProRealTime non propone servizi di consiglio su investimenti. Gli esempi presentati in questo manuale sono a scopo pedagogico. Tutte le informazioni presenti sono a carattere generale e non rappresentano in nessun caso informazioni o consulenze personalizzate o incitazioni a comprare o vendere strumenti finanziari. Le performances passate non rappresentano previsioni per il futuro. Ogni sistema di trading può esporre a un rischio di perdite superiori all'investimento iniziale.

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 programmazione.

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 possibilità 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, assimilerete 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, oppure 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.

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.

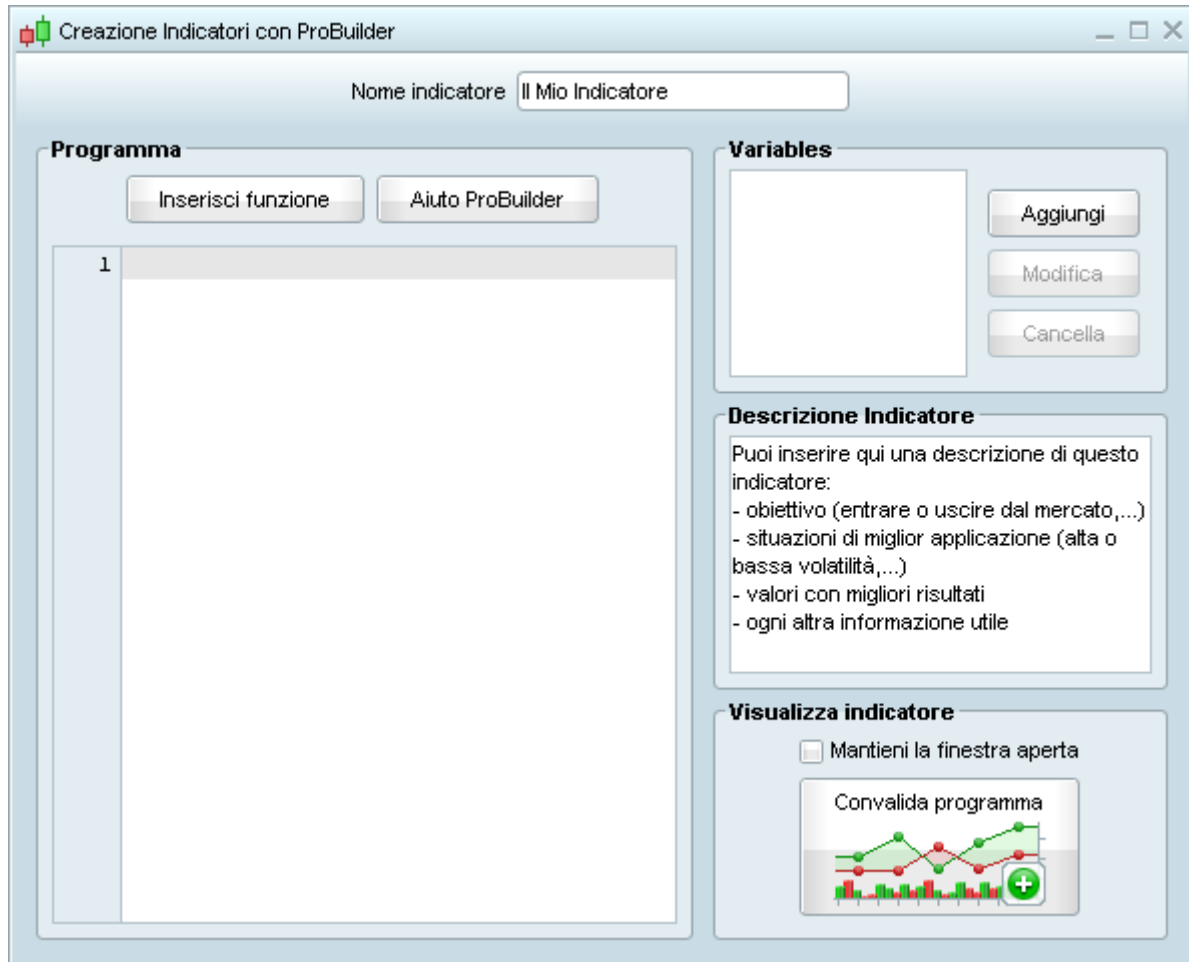


Accederete dunque alla finestra di gestione degli indicatori, dove potrete:

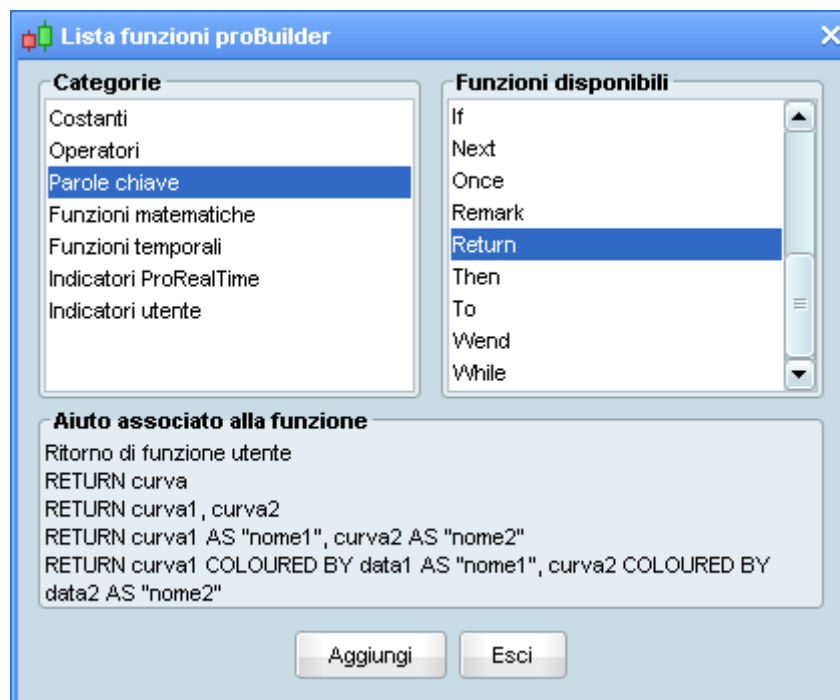
- Visualizzare un indicatore predefinito
- Creare un indicatore personalizzato che potrà in seguito essere applicato a un qualsiasi valore

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

- Programmare direttamente 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, cioè 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.

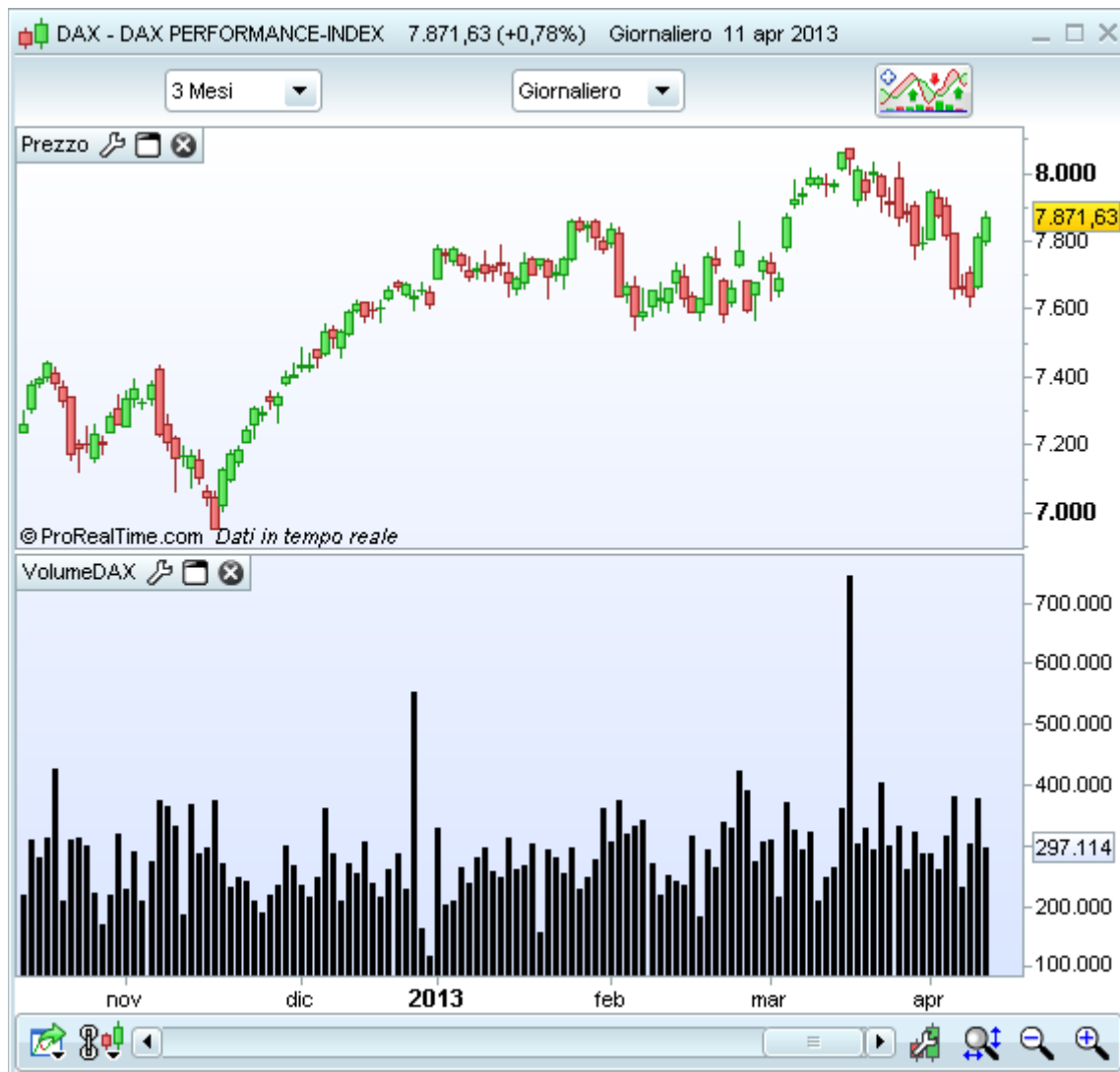


RETURN vi permette di visualizzare il risultato come indicatore

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". Non dimenticate di aggiungere uno spazio tra ogni comando, come illustrato di seguito.



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 / Decomenta (Ctrl + R): Commenta il codice selezionato / Decomenta 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'è differenza tra minuscole e maiuscole
- 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 "vARiABlE", 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:

X ← 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 con il segno =

X = Volume

Il simbolo = é utilizzato:

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

Le costanti finanziarie ProBuilder

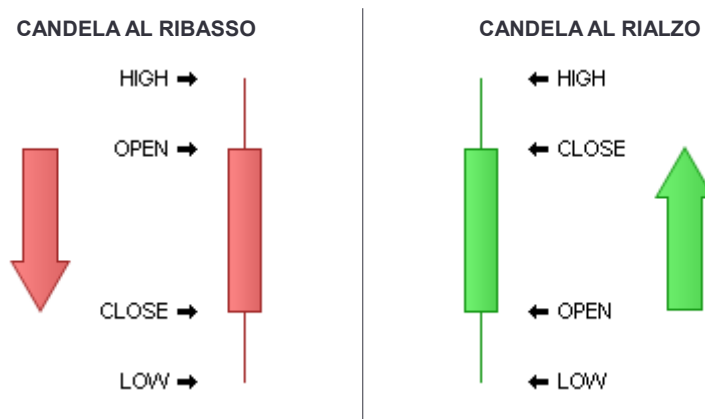
Prima di cominciare a programmare i vostri indicatori personali, è necessario passare in rassegna gli elementi a partire dai quali potrete costruire il vostro codice, quali i prezzi di apertura, di chiusura, il volume etc...

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 costanti di prezzo e di volume che si adattano al timeframe del grafico

Si tratta delle costanti classiche più utilizzate. Riportano per default 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, è 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 presente alla barra in corso sarà chiamato da: Open[2].

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

Le costanti giornaliere di prezzo

Contrariamente costanti di prezzo e di volume che si adattano al timeframe del grafico, le costanti 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 'n' è uguale a 0, n fa riferimento al giorno in corso. Poiché i valori High e Low non sono definiti per n=0, otteniamo un risultato per i giorni precedenti ma non per il giorno corrente.



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

`Close[3]` ➔ Il prezzo di chiusura 3 periodi fa

`Dclose(3)` ➔ Il prezzo di chiusure 3 giorni fa

Le costanti temporali

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

I traders ne conoscono però 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 2020 dunque per il programma risulta 20200704.

La data può essere letta anche nel modo seguente

20200704 = 2020 anni 07 mesi e 4 giorni

Attenzione: MM non può essere superiore a 12 e JJ non può essere superiore a 31.

- **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)
```

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=lunedì, 2=martedì, 3=mercoledì, 4=giovedì, 5=venerdì

Esempio di impiego delle istruzioni:

```
a = (Hour > 17)
```

```
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](#)

(Attenzione a non confondere "Day" e "Days")

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

Il contatore parte da sinistra a destra e conta ogni barra dalla 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 **BarIndex**, 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 partecipanti.

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 costituita da queste 5 costanti permette dunque di portare una spiegazione un po' 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 default 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 è:

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 default posizionati per il **periodo** e per l'argomento del **prezzo o dell'indicatore**.

Average[20](Close)

I valori possono essere modificati. Per esempio, è possibile sostituire le 20 barre definite come periodo, con un altro qualsiasi valore di barre (es: Average[10], Average[15], Average[20]... Parimenti, è possibile rimpiazzare "Close" con "Open" o RSI (Relative Strength Index). Questo darebbe per esempio:

Average[20](RSI[5])

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.

```
// Calcolo della linea MACD
LineaMACD = ExponentialAverage[12](Close) - ExponentialAverage[26](Close)
// Calcolo della linea del Segnale MACD
LineaSegnale = ExponentialAverage[9](LineaMACD)
// 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 default 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 modificare i parametri dell'indicatore senza entrare nel codice.

Calcoliamo per esempio una media mobile di periodo 20:

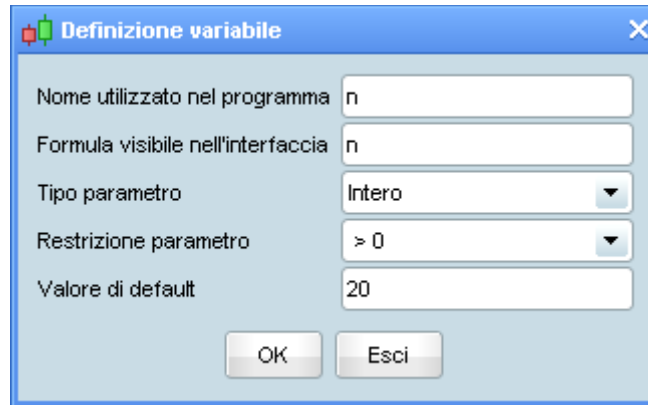
```
RETURN Average[20](Close)
```



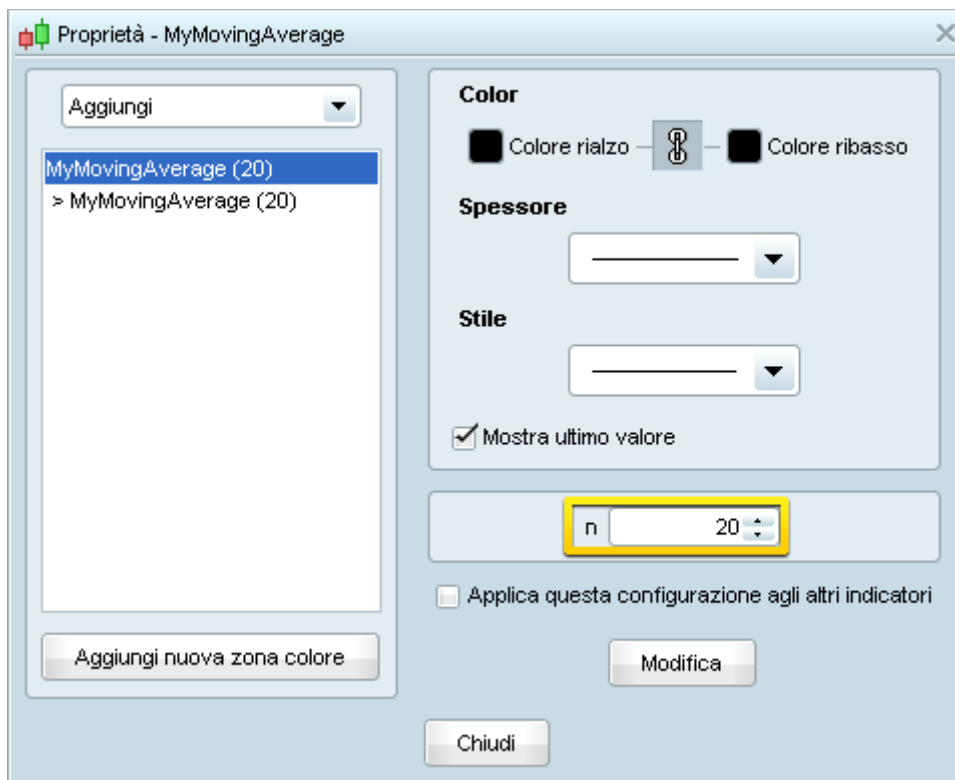
Per poter modificare il numero di periodi utilizzati per il calcolo, basterà sostituire il numero 20 con la variabile "n":

```
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 così modificata



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

Capitolo II: Funzioni matematiche e istruzioni ProBuilder

Strutture di controllo

Istruzioni condizionata IF

L'istruzione **IF** serve ad effettuare una scelta di azione condizionata, cioè 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.

Una 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 default sin rinvia 0).

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

<code>Result = 0</code>	Result è uguale a 0
<code>IF Close > Average[20](Close) THEN</code>	Se il prezzo di chiusura > alla media mobile di 20 periodi ALLORA
<code>Result = 1</code>	Result sarà uguale a 1, altrimenti Result = 0
<code>ENDIF</code>	FINE CONDIZIONE
<code>RETURN Result</code>	



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, cioè 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' possibile 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:

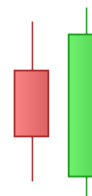
```
IF (Condizione1) THEN
  (Azione1)
ELSIF (Condizione2) THEN
  (Azione2)
ELSIF (Condizione3) THEN
  (Azione3)
...
...
...
ELSE
  (Azione n)
ENDIF
```

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 capitolo 1 di questo manual, **BarIndex** è stato presentato come un contatore del numero di barre dall'inizio dello storico illustrato. **BarIndex** è spesso utilizzato in associazione con **IF**. Per esempio, vogliamo 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** è utilizzata quando si desidera richiamare uno per uno, una serie finita di elementi (es: 1, 2, 3, ...6, 7 o 7, 6, ... 3, 2, 1). Questi ultimi potranno essere di qualsiasi tipo, se la serie è 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 decrescente degli elementi da considerare. È 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

Questo codice significa che "Variabile" prenderà in considerazione in un primo momento il valore di inizio della serie, poi prenderà il valore seguente (il precedente +1) e così via, fino a giungere ad un valore di variabile uguale o superiore all'ultimo numero della serie. Chiudiamo così la serie.

Esempio: Media mobile sulle ultime 20 barre sui massimi

SUMhigh = 0	
IF BarIndex < 5 THEN	Se non ci sono ancora 5 periodi sullo storico
MMhigh = Undefined	Attribuiamo a MMhigh il valore "nullo"
ELSE	Altrimenti
FOR i = 0 TO 4 DO	Per i valori compresi tra 1 e 5
SUMhigh = High[i]+SUMhigh	Sommiamo gli ultimi 5 valori dei massimi
NEXT	
ENDIF	
MMhigh = SUMhigh / 5	Facciamo la media degli ultimi 5 periodi e attribuiamo il risultato alla MMhigh
RETURN MMhigh	Visualizziamo MMhigh

Ordine decrescente (FOR, DOWNTO, DO, NEXT)

L'ordine decrescente utilizza le seguenti 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 5 ultime barre sui massimi:

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

```
IF BarIndex = 0 THEN
    MMhigh = Undefined
ELSE
    FOR i = 4 DOWNTO 0 DO
        SUMhigh = High[i] + SUMhigh
    NEXT
ENDIF
MMhigh = SUMhigh / 5
RETURN MMhigh
```

L'istruzione WHILE

WHILE serve ad applicare una condizione finché questa è 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
```

Questo codice consente di visualizzare il numero di barre che separano la barra corrente con il precedente massimo nel limite di 30 periodi:

```
i = 1
WHILE high > high [i] and i < 30 DO
    i = i + 1
WEND
RETURN i
```

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 è 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 può testare se *i* è uguale a 10.

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

Il codice corretto è invece:

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

In questo codice, *i* è 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, è possibile combinare **BREAK** con l'istruzione **IF**.

Con WHILE

Quando cerchiamo di uscire da un'istruzione **WHILE**, utilizziamo **BREAK** il giorno successivo:

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

L'uso di **BREAK** in un'istruzione **WHILE** è interessante solo se vogliamo testare una condizione supplementare per cui il valore non può essere rilevato all'interno dell'istruzione **WHILE**. Per esempio guardiamo questo indicatore stocastico che è calcolato solo in una tendenza al rialzo:

```
Linea = 0
Aumento = (Close - Close[1]) > 0
i = 0
WHILE Aumento[i] DO
    i = i + 1
    // Se High - low, usciamo dal circolo per evitare una divisione per zero.
    IF (high-low) = 0 then
        BREAK
    ENDIF
    osc = (close - low) / (high - low)
    Linea = AVERAGE [i] (osc)
WEND
RETURN Linea
```

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 è 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, riletture incluse.
- In caso di riletture del codice da parte del linguaggio, essa conserverà i valori calcolati durante la lettura precedente.

Per meglio comprendere l'utilizzo della funzione ONCE, è 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 è 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**", ricomincerà 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 così' via fino a (Count = 15; i = 6). Arrivati a questo risultato, l'istruzione in **IF** non è più trattata poiché la condizione non è 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 è strettamente inferiore a b
- **$a <= b$ o $a =< b$** : a è inferiore o uguale a b
- **$a > b$** : a è strettamente superiore a b
- **$a >= b$ o $a => b$** : a è superiore o uguale a b
- **$a = b$** : a è uguale a b (o riceve il valore b)
- **$a <> b$** : a è 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, cioè:

```
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 è 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 così ignorate dal programma. Facciamo un esempio di inserimento commenti:

```
REM questo programma rinvia la media mobile aritmetica 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 – questo non è applicabile alle sezioni **REM** (commenti)

CustomClose

CustomClose è 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 è la stessa delle costanti di prezzo 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 è possibile configurare i prezzi presi in considerazione per il calcolo.

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 "Weighted Moving Average", c COLOURED(0, 0, 255) AS "Simple Moving Average"
```

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



Istruzioni di disegno

Le seguenti istruzioni sono state aggiunte nella versione 10.3 del software. Sono disponibili su questa versione e successive. Queste istruzioni consentono di disegnare oggetti sui grafici e di personalizzare il disegno delle candele e delle barre nei grafici.

- **BACKGROUNDCOLOR**: questa funzione consente di colorare il fondo del grafico per una candela (come gioni pari/dispari). La zona di colore comincia a metà strada tra la barra precedente e la successiva.

Esempio: `BACKGROUNDCOLOR (0, 127, 255, 25)`

È possibile utilizzare una variabile per i colori, se si desidera modificare il colore di sfondo in base alle proprie condizioni.

Esempio: `BACKGROUNDCOLOR (0, color, 255, 25)`



- **DRAWBARCHART**: disegna una barra sul grafico. Apertura, chiusura, minimo e massimo possono essere costanti o variabili.

Esempio: `DRAWBARCHART (open, high, low, close) COLOURED (0, 255, 0)`

- **DRAWCANDLE**: disegna una candela sul grafico. Apertura, chiusura, minimo e massimo possono essere costanti o variabili.

Esempio: `DRAWCANDLE (open, high, low, close) COLOURED (0, 255, 0)`



- **DRAWARROW**: disegna una freccia con punta verso un punto definito. Bisogna definire un punto per la freccia (scala orizzontale e verticale). Si può scegliere il colore.

Esempio: `DRAWARROW (x1, y1) COLOURED (R, V, B, a)`

- **DRAWARROWUP**: disegna una freccia con punta verso l'alto. Bisogna definire un punto per la freccia (scala orizzontale e verticale). Si può scegliere il colore.

Esempio: `DRAWARROWUP (x1, y1) COLOURED (R, V, B, a)`

This is useful to add visual buy signals.

- **DRAWARROWDOWN**: disegna una freccia con punta verso il basso. Bisogna definire un punto per la freccia (scala orizzontale e verticale). Si può scegliere il colore.

Esempio: `DRAWARROWDOWN (x1, y1) COLOURED (R, V, B, a)`

Questo è utile per aggiungere segnali visivi di vendita.



- **DRAWRECTANGLE**: disegna un rettangolo nel grafico.

Esempio: `DRAWRECTANGLE (x1, y1, x2, y2) COLOURED (R, V, B, a)`

- **DRAWELLIPSE**: disegna un'ellisse sul grafico.

Esempio: `DRAWELLIPSE (x1, y1, x2, y2) COLOURED (R, V, B, a)`



- **DRAWLINE**: disegna una linea sul grafico.

Esempio: `DRAWLINE (x1, y1, x2, y2) COLOURED (R, V, B, a)`

- **DRAWHLINE**: disegna una linea orizzontale sul grafico.

Esempio: `DRAWHLINE (y1) COLOURED (R, V, B, a)`

- **DRAWVLINE**: disegna una linea verticale sul grafico.

Esempio: `DRAWVLINE (x1) COLOURED (R, V, B, a)`

- **DRAWSEGMENT**: disegna un segmento sul grafico.

Esempio: `DRAWSEGMENT (x1, y1, x2, y2) COLOURED (R, V, B, a)`



- **DRAWTEXT**: aggiunge una sezione testo nel grafico con il testo di sua scelta in un punto specifico.

Esempio: `DRAWTEXT ("your text", x1, y1) COLOURED (R, V, B, a)`



- **DRAWONLASTBARONLY**: funzione che consente di disegnare oggetti disegnati solamente sull'ultima candela. Questa funzione dovrebbe essere sempre utilizzata con `"CALCULATEONLASTBARS"` per ottimizzare i calcoli.

Esempio: `DEFPARAM DRAWONLASTBARONLY = true`

- **CALCULATEONLASTBARS**: consente di aumentare la velocità alla quale un indicatore viene calcolato, definendo il numero di candele che presentano il risultato.

Esempio: `DEFPARAM CALCULATEONLASTBARS = 200`

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 continuare 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 ternario: 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, cioè 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, cioè 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, cioè 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 = AmplitudeUp * Price
    StopSHORT = AmplitudeDown * 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, cioè 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, cioè 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 grafico 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 DOWNT0 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
- $[(\text{valore assoluto dell'apertura del giorno} - \text{chiusura del giorno precedente}) / \text{chiusura del giorno precedente}]$ 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"
```

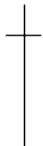
- Doji (versione larga)



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:

$$\text{BodyMomentum} = 100 * \text{BodyUp} / (\text{BodyUp} + \text{BodyDown})$$

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'oscillatore 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 è simile allo Stocastico. Per tracciarlo, è necessario tracciare innanzitutto due curve. Il %R è 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 è 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)	Designa l'accumulazione distribuzione classica
ADX	ADX[N]	Indicatore Average Directional Index di n periodi
ADXR	ADXR[N]	Indicatore Average Directional Index Rate di n periodi
AND	a AND b	Operatore logico E
AroonDown	AroonDown[P]	Designa l'Aroon Down di n periodi
AroonUp	AroonUp[P]	Designa l'Aroon Up di n periodi
ATAN	ATAN(a)	Funzione matematica Arco Tangente
AS	RETURN x AS "ResultName"	Istruzione che serve ad attribuire un nome ad una linea o indicatore mostrato sul grafico. Usato con "RETURN"
Average	Average[N](price)	Media Mobile Semplice di n periodi
AverageTrueRange	AverageTrueRange[N](price)	Media mobile lisciata da Wilder

B

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
BACKGROUNDCOLOR	BACKGROUNDCOLOR(R,V,B ,a)	Consente di colorare il fondo del grafico o per una candela
BarIndex	BarIndex	Numero di barre dall'inizio dei dati caricati (per un sistema di trading nel caso di ProBacktest o ProOrder, o in un grafico nel caso di un indicatore ProBuilder). Vedi anche: PreLoadBars.
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
<code>CALCULATEONLASTBARS</code>	DEFPARAM CalculateOnLastBars = 200	Consente di aumentare la velocità alla quale un indicatore viene calcolato, definendo il numero di candele che presentano il risultato.
<code>CALL</code>	myResult=CALL myFunction	Permette di richiamare un altro indicatore
<code>CCI</code>	CCI[N](price) o CCI[N]	Dà il Commodity Channel Index
<code>ChaikinOsc</code>	ChaikinOsc[Ch1, Ch2](price)	Designa l'oscillatore di Chaikin
<code>Chandle</code>	Chandle[N](price)	Designa il Chande Momentum Oscillator
<code>ChandeKrollStopUp</code>	ChandeKrollStopUp[Pp, Qq, X]	Stop di protezione secondo Chande e Kroll in posizione di acquisto
<code>ChandeKrollStopDown</code>	ChandeKrollStopDown[Pp, Qq, X]	Stop di protezione secondo Chande e Kroll in posizione di vendita
<code>Close</code>	Close[N]	Designa il prezzo di chiusura della barra corrente oppure l'ultimo prezzo registrato
<code>COLOURED</code>	RETURN x COLOURED(R,G,B)	Permette di personalizzare il colore di una curva, secondo la convenzione RGB
<code>COS</code>	COS(a)	Funzione coseno
<code>CROSSES OVER</code>	a CROSSES OVER b	Operatore booleano di incrocio a rialzo
<code>CROSSES UNDER</code>	a CROSSES UNDER b	Operatore booleano di incrocio a ribasso
<code>cumsum</code>	cumsum(price)	Somma di un prezzo dall'inizio dello storico mostrato
<code>CurrentDayOfWeek</code>	CurrentDayOfWeek	Designa il giorno attuale
<code>CurrentHour</code>	CurrentHour	Designa l'ora attuale
<code>CurrentMinute</code>	CurrentMinute	Designa il minuto attuale
<code>CurrentMonth</code>	CurrentMonth	Designa il mese attuale
<code>CurrentSecond</code>	CurrentSecond	Designa il secondo attuale
<code>CurrentTime</code>	CurrentTime	Designa l'ora minuto attuale
<code>CurrentYear</code>	CurrentYear	Designa l'anno attuale
<code>CustomClose</code>	CustomClose[N]	Costante parametrabile nella finestra proprietà
<code>Cycle</code>	Cycle(price)	Indicatore Cycle

D

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
Date	Date[N]	Designa la data di ogni barra caricata nel grafico
Day	Day[N]	Designa il giorno di ogni barra caricata nel grafico
Days	Days[N]	Contatore di giorni dal 1900
DayOfWeek	DayOfWeek[N]	Designa il giorno della settimana di ogni barra
DClose	DClose(N)	Prezzo di chiusura dell'ennesima giornata anteriore a quella della barra corrente
DEMA	DEMA[N](price)	Doppia Media Mobile Esponenziale
DHigh	DHigh(N)	Prezzo più alto dell'ennesima giornata precedente a quella della barra corrente
DI	DI[N](price)	Designa il DI+ meno il DI-
DIminus	DIminus[N](price)	Designa l'indicatore DI-
Dlplus	Dlplus[N](price)	Designa l'indicatore DI+
DLow	DLow(N)	Il Minimo dell'ennesima giornata anteriore a quella della barra corrente
DO	Vedere FOR e WHILE	Istruzione facoltativa dei FOR e WHILE per l'azione di chiusura
DOpen	DOpen(N)	Prezzo di apertura dell'ennesima giornata anteriore a quella della barra corrente
DOWNTO	Vedere FOR	Istruzione di lettura decrescente per FOR
DPO	DPO[N](price)	Designa il Detrended Price Oscillator
DRAWARROW	DRAWARROW(x1,y1)	Disegna una freccia con punta verso il punto selezionato. Questa funzione e le successive sono disponibili solo per la versione 10.3 e successive
DRAWARROWDOWN	DRAWARROWDOWN(x1,y1)	Disegna una freccia con punta verso il basso
DRAWARROWUP	DRAWARROWUP(x1,y1)	Disegna una freccia con punta verso l'alto
DRAWBARCHART	DRAWBARCHART(open,high,low,close)	Disegna una barra sul grafico. Apertura, chiusura, minimo e massimo possono essere costanti o variabili
DRAWCANDLE	DRAWCANDLE(open,high,low,close)	Disegna una candela sul grafico. Apertura, chiusura, minimo e massimo possono essere costanti o variabili
DRAWELLIPSE	DRAWELLIPSE(x1,y1,x2,y2)	Disegna un'ellisse sul grafico
DRAWHLINE	DRAWHLINE(y1)	Disegna una linea orizzontale sul grafico
DRAWLINE	DRAWLINE(x1,y1,x2,y2)	Disegna una linea sul grafico
DRAWONLASTBARONLY	DEFPARAM DrawOnLastBarOnly = true	Consente di disegnare oggetti disegnati solamente sull'ultima candela

<code>DRAWRECTANGLE</code>	<code>DRAWRECTANGLE(x1,y1,x2,y2)</code>	Disegna un rettangolo nel grafico
<code>DRAWSEGMENT</code>	<code>DRAWSEGMENT(x1,y1,x2,y2)</code>	Disegna un segmento sul grafico
<code>DRAWTEXT</code>	<code>DRAWTEXT("your text", x1,y1)</code>	Aggiunge una sezione testo nel grafico con il testo di sua scelta in un punto specifico
<code>DRAWVLINE</code>	<code>DRAWVLINE(x1)</code>	Disegna una linea verticale sul grafico

E

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
<code>EaseOfMovement</code>	<code>EaseOfMovement[I]</code>	Designa l'indicatore Ease of Movement
<code>ELSE</code>	Vedere IF/THEN/ELSE/ENDIF	Istruzione per introdurre la seconda condizione in alternativa alla prima uscita di IF
<code>ELSEIF</code>	Vedere IF/THEN/ELSEIF/ELSE/ENDIF	Contrazione di ELSE IF (da inserire all'interno della condizione)
<code>EMV</code>	<code>EMV[N]</code>	Designa l'indicatore Ease of Movement Value
<code>ENDIF</code>	Vedere IF/THEN/ELSE/ENDIF	Istruzione di chiusura delle istruzioni condizionali
<code>EndPointAverage</code>	<code>EndPointAverage[N](price)</code>	Media Mobile all'ultimo punto
<code>EXP</code>	<code>EXP(a)</code>	Funzione matematica esponenziale
<code>ExponentialAverage</code>	<code>ExponentialAverage[N](price)</code>	Media Mobile Esponenziale

F - G

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
<code>FOR/TO/NEXT</code>	<code>FOR i=a TO b DO a NEXT</code>	Ciclo FOR (elabora tutti i valori con ordini crescenti (TO) o decrescenti (DOWNT0))
<code>ForceIndex</code>	<code>ForceIndex(price)</code>	Indicatore Force Index che determina chi controlla il mercato (compratore o venditore)

H

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
<code>High</code>	<code>High[N]</code>	Designa il massimo della barra corrente o quella di n barre precedenti
<code>highest</code>	<code>highest[N](price)</code>	Designa il prezzo massimo di un insieme di barre
<code>HistoricVolatility</code>	<code>HistoricVolatility[N](price)</code>	Designa la volatilità storica o statistica
<code>Hour</code>	<code>Hour[N]</code>	Designa l'ora di chiusura di ogni barra

I - J - K

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
IF/THEN/ENDIF	IF a THEN b ENDIF	Insieme di istruzioni condizionali senza la seconda condizione
IF/THEN/ELSE/ENDIF	IF a THEN b ELSE c ENDIF	Insieme di istruzioni condizionali
IntradayBarIndex	IntradayBarIndex[N]	Conta il numero di candele sul grafico intraday

L

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
LinearRegression	LinearRegression[N](price)	Indicatore di regressione lineare
LinearRegressionSlope	LinearRegressionSlope[N](price)	Inclinazione di regressione lineare
LOG	LOG(a)	Funzione matematica logaritmica
Low	Low[N]	Designa il minimo della barra corrente o quella di n barre precedenti
lowest	lowest[N](price)	Designa il minimo di un periodo su un insieme di barre

M

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
MACD	MACD[S,L,Si](price)	Moving Average Convergence Divergence (MACD)
MACDline	MACDLine[S,L](price)	Designa la linea del MACD
MassIndex	MassIndex[N]	Indicatore Mass Index applicato su N barre
MAX	MAX(a,b)	Funzione matematica "Massimo"
MedianPrice	MedianPrice	Media del prezzo massimo e minimo
MIN	MIN(a,b)	Funzione matematica "Minimo"
Minute	Minute	Designa il minuto di ogni barra caricata nel grafico
MOD	a MOD b	Funzione matematica "Resto della divisione"
Momentum	Momentum[I]	Momentum (prezzo di chiusura – prezzo della chiusura dell'ennesima barra precedente)
MoneyFlow	MoneyFlow[N](price)	Dà il MoneyFlow tra -1 e 1
MoneyFlowIndex	MoneyFlowIndex[N]	Designa il MoneyFlowIndex
Month	Month[N]	Designa il mese di chiusura di ogni barra caricata nel grafico

N

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
NEXT	Vedere FOR/TO/NEXT	Istruzione da posizionare alla fine del ciclo "FOR"
NOT	Not A	Operatore logico NON

O

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
OBV	OBV(price)	Designa l' "On-Balance-Volume"
ONCE	ONCE VariableName = VariableValue	Istruzione che si vuole realizzare una sola volta
Open	Open[N]	Designa il prezzo di apertura della barra corrente o quella di n barre precedenti
OR	a OR b	Operatore logico O

P - Q

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
PIPSIZE	PipSize	Grandezza di un pip (o punto), PipSize=PointSize
PriceOscillator	PriceOscillator[S,L](price)	Indicatore Percertage Price oscillator
PositiveVolumeIndex	PriceVolumeIndex(price)	Designa l'indicatore Positive Volume Index
PVT	PVT(price)	Designa l'indicatore "Price Volume Trend"

R

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
R2	R2[N](price)	Coefficiente della radice quadrata (tasso di errore della regressione lineare sul prezzo)
Range	Range[N]	Calcola il Range (Differenza tra il massimo e il minimo)
REM	REM comment	Introduce un commento (non preso in considerazione nel codice)
Repulse	Repulse[N](price)	Indicatore Repulse (misura la spinta al rialzo e al ribasso di ogni candela)
RETURN	RETURN Result	Istruzione che ritorna il risultato
ROC	ROC[N](price)	Designa il "Price Rate of Change"
ROUND	ROUND(a)	Funzione matematica "Arrotondamento all'unità"
RSI	RSI[N](price)	Designa l'oscillatore "Relative Strength Index"

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	YYYYMMDD	Data 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)	Calcola 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
<code>Year</code>	<code>Year[N]</code>	Permette di richiamare un anno in particolare nel programma
<code>Yesterday</code>	<code>Yesterday[N]</code>	Permette di richiamare il giorno precedente nel programma

Z

CODICE	IMPLEMENTAZIONE	DESCRIZIONE
<code>ZigZag</code>	<code>ZigZag[Zr](price)</code>	Zig-Zag della teoria delle onde di Elliot
<code>ZigZagPoint</code>	<code>ZigZagPoint[Zp](price)</code>	Zig-Zag della teoria delle onde di Elliot calcolata a Zp punti

Operatori

CODICE	DESCRIZIONE	CODICE	DESCRIZIONE
<code>+</code>	Somma	<code><></code>	Differenza
<code>-</code>	Sottrazione	<code><</code>	Strettamente inferiore
<code>*</code>	Moltiplicazione	<code>></code>	Strettamente superiore
<code>/</code>	Divisione	<code><=</code>	Inferiore
<code>=</code>	Uguale	<code>>=</code>	Superiore

ProRealTime SOFTWARE

