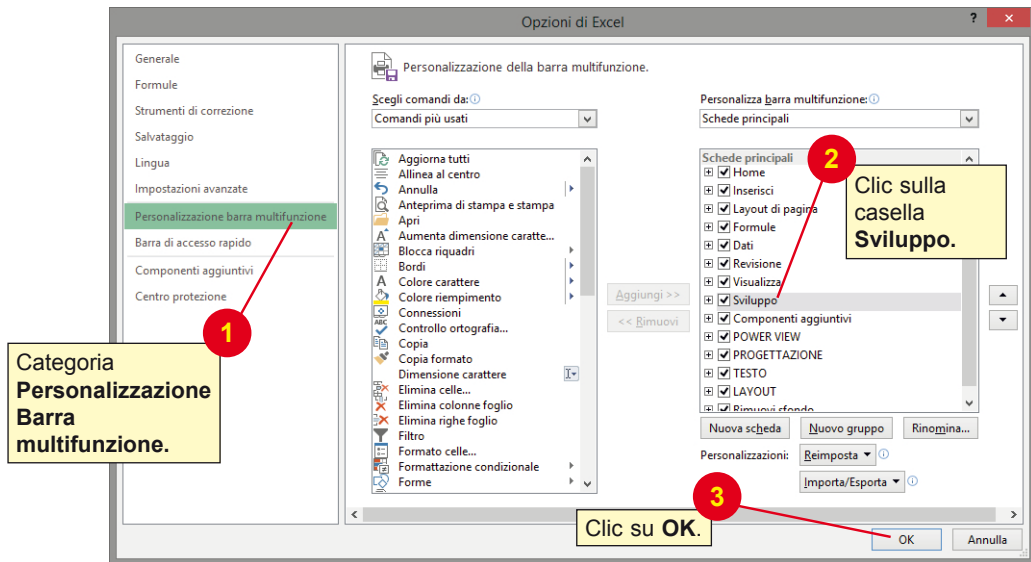


Note per la programmazione in linguaggio Visual Basic di Excel

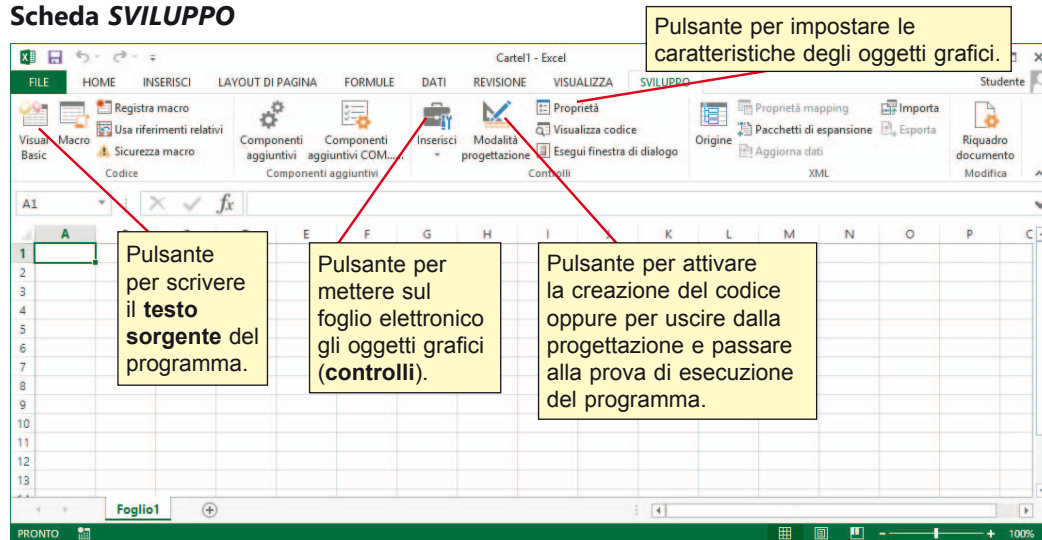
L'ambiente di programmazione

Il foglio elettronico *Excel*, così come gli altri prodotti *Office* di Microsoft, possiede un vero e proprio ambiente di programmazione in linguaggio **Visual Basic**, che consente di creare procedure software dalla fase di editing del testo sorgente fino all'esecuzione del programma.

Per attivare questo ambiente si deve utilizzare la scheda **SVILUPPO**, che non è compresa tra le schede standard della Barra multifunzione e quindi deve essere aggiunta dalla finestra **Opzioni di Excel** che si apre facendo clic sul menu **FILE** e scegliendo **Opzioni**.



Scheda SVILUPPO



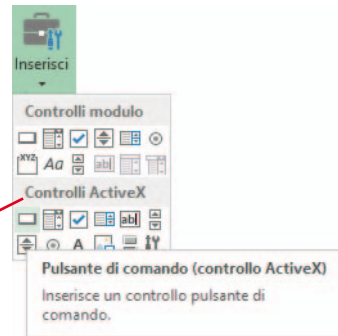
I file di Excel che contengono codice Visual Basic devono essere salvati come **Cartella di lavoro con attivazione macro di Excel** (casella **Salva come** nella finestra **Salva con nome**): questo tipo di file ha estensione **.xlsm**.

Le **fasi di lavoro** di una tipica sessione di programmazione in Visual Basic sono:

1 Clic sul pulsante **Modalità progettazione**.

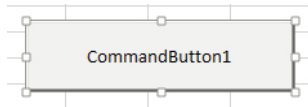


2 Clic sul pulsante **Inserisci** e, nell'elenco **Controlli ActiveX**, scegliere un controllo grafico, per esempio un **Pulsante di comando** (passando con il mouse sopra gli oggetti si può evidenziare il nome dell'oggetto).



La casella dei controlli consente di scegliere tra i **controlli grafici** standard che si possono inserire in un progetto Visual Basic per costruire l'interfaccia con l'utente. Gli strumenti grafici sono: caselle di testo, etichette, pulsanti di comando, liste, ecc., cioè tutti gli oggetti tipici dell'interfaccia grafica di Windows.

3 Disegnare l'oggetto sul foglio Excel trascinando il mouse e usando i quadratini di ridimensionamento.

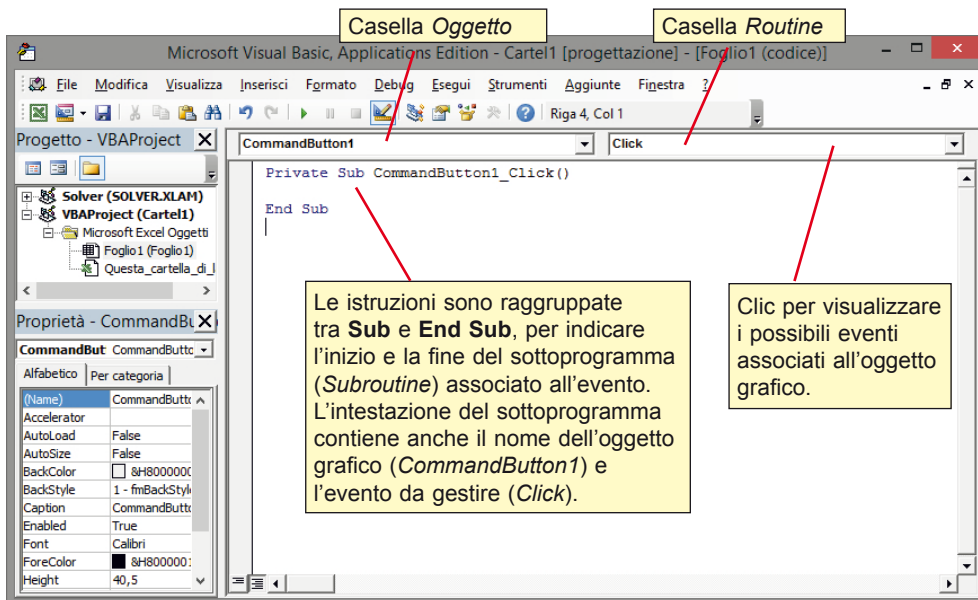



4 Doppio clic sull'oggetto per attivare l'editor del codice.

5 Nella **finestra del codice** che si apre scrivere le istruzioni in linguaggio Visual Basic che devono essere eseguite quando accade un **evento** sull'oggetto.

Un **evento** è tipicamente il clic dell'utente sull'oggetto (*Click*), ma può essere anche un doppio clic (*DoubleClick*) o un passaggio del mouse sopra di esso (*MouseMove*).

In sostanza il codice specifica che cosa deve fare il computer come risposta alle azioni dell'utente.



6 Tornare al foglio Excel facendo clic sulla prima icona a sinistra nella barra degli strumenti della finestra del codice (l'icona con il logo di Excel)  oppure facendo clic sul pulsante del foglio Excel nella Barra delle applicazioni nella parte inferiore del desktop.

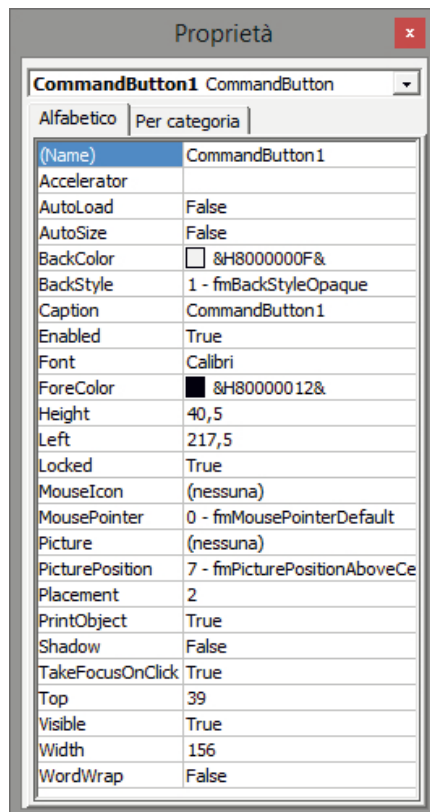
7 Uscire dalla modalità progettazione facendo clic sul pulsante **Modalità progettazione**.

8 Provare l'esecuzione del codice facendo clic sull'oggetto grafico.

MODALITÀ ALTERNATIVA 1 Per passare dal foglio di Excel alla finestra del codice Visual Basic, e viceversa: scorciatoia da tastiera con i tasti **Alt + F11**.

MODALITÀ ALTERNATIVA 2 Per aprire la finestra del codice Visual Basic: clic con il tasto destro del mouse sopra un oggetto grafico e scelta **Visualizza codice** dal menu di scelta rapida. Dal codice si può poi tornare all'oggetto con la combinazione di tasti **Maiuscolo + F7**.

Ogni oggetto possiede alcune **proprietà** che possono essere impostate dal programmatore, se non vuole mantenere quelle prefissate (*proprietà di default*): esse riguardano la forma, il colore, il nome, la disposizione dell'oggetto.



L'elenco delle proprietà di un oggetto è visualizzato facendo clic su di esso con il tasto destro del mouse e scegliendo **Proprietà** dal menu di scelta rapida.

MODALITÀ ALTERNATIVA Clic sul pulsante **Proprietà** nel gruppo **Controlli** della scheda **Sviluppo**.

ESEMPIO

Calcolare l'ipotenusa di un triangolo rettangolo, noti i cateti.

- *Dati di input:* le misure dei due cateti, c_1 , c_2 .
- *Dati di output:* la misura dell'ipotenusa $ipot$.
- *Risoluzione:* dopo aver acquisito i valori di c_1 e c_2 , si calcola l'ipotenusa $ipot$ con la formula

$$ipot = \sqrt{c_1^2 + c_2^2}$$

Algoritmo TriangoloRettangolo

variabili

dichiara c_1 , c_2 ,
 $ipot$ come numeri reali

inizio

immetti c_1
immetti c_2
assegna $ipot = \sqrt{c_1^2 + c_2^2}$
scrivi $ipot$

fine

riga di intestazione

sezione dichiarativa

sezione esecutiva

Interfaccia utente (*TriangoloRettangolo.xlsm*)

Definiamo la disposizione dei dati e degli oggetti grafici sul foglio elettronico (*layout*).

Attivazione della Modalità progettazione.

Selezione del pulsante di comando dalla casella dei Controlli ActiveX del pulsante Inserisci.

Clic con il tasto destro del mouse: scelta Proprietà dal menu di scelta rapida.

Cambiare la proprietà Caption, che rappresenta il testo che compare sopra il pulsante di comando (Calcola ipotenusa).

Codice

Associamo ora al pulsante di comando il codice del sottoprogramma che deve essere eseguito quando l'utente fa un clic sul pulsante di comando.

Per passare alla finestra del codice basta fare un doppio clic sul pulsante di comando oppure premere la combinazione dei tasti **Alt + F11** dopo aver selezionato l'oggetto grafico.

Il codice che realizza il calcolo dell'ipotenusa è il seguente:

```
Private Sub CommandButton1_Click()  
  ' dichiarazione delle variabili  
  Dim c1, c2, ipot As Single  
  ' acquisizione delle misure dei cateti  
  c1 = Range("B1")  
  c2 = Range("B2")  
  ' calcolo dell'ipotenusa  
  ipot = Sqr(c1 ^ 2 + c2 ^ 2)  
  ' visualizzazione del risultato  
  Range("B3") = ipot  
End Sub
```

Sottoprogramma (**Sub**) che viene eseguito quando l'utente provoca l'evento **Click** sopra il controllo grafico **CommandButton1**.

Dim (dimensione) elenca i nomi delle variabili utilizzate e specifica il tipo con la clausola **As**; il tipo **Single** rappresenta i numeri reali in singola precisione.

Range (intervallo) specifica la cella o l'intervallo di celle del foglio elettronico che contengono i dati.

Le frasi che iniziano con l'apice sono **frasi di commento** che servono a documentare le istruzioni del programma.

Gli **operatori aritmetici** sono **+**, **-**, *****, **/** per le quattro operazioni fondamentali (addizione, sottrazione, moltiplicazione e divisione), **^** per l'elevamento a potenza, **** per il quoziente della divisione intera, **MOD** per il resto della divisione intera.

Sqr è la funzione predefinita del linguaggio che calcola la radice quadrata dell'espressione indicata tra le parentesi tonde.

L'**assegnazione** dei valori alle variabili o ad un intervallo (indicata con il segno =) è sempre verso sinistra.

Per esempio, l'istruzione

```
c1 = Range("B1")
```

indica che il valore della cella B1 del foglio elettronico è assegnato alla variabile *c1*, mentre nell'istruzione

```
Range("B3") = ipot
```

il valore della variabile *ipot* è assegnato alla cella B3 del foglio elettronico.

Nel testo sorgente del programma le parole chiave del linguaggio sono scritte in **azzurro** e le frasi di commento sono in colore **verde**. Inoltre le eventuali righe di codice contenenti errori sono evidenziate in **rosso**.

Proviamo ora il funzionamento del sottoprogramma. Torniamo al foglio elettronico (con la combinazione di tasti **Alt + F11** oppure con un clic sul pulsante del foglio Excel nella barra delle applicazioni) e disattiviamo la *Modalità progettazione* facendo clic sull'apposito pulsante della scheda **Sviluppo**.

Facendo clic sul pulsante di comando con la scritta *Calcola ipotenusa*, il sottoprogramma viene eseguito e nella cella B3 compare il valore calcolato dell'ipotenusa.

Variabili e costanti

La dichiarazione delle variabili è indicata con la parola chiave **Dim**. Se si scrivono solo i nomi delle variabili, senza specificare il tipo di dato che esse rappresentano (numero intero o reale, oppure stringa), il tipo di dato viene determinato implicitamente dal programma sulla base del dato che viene letto nelle celle del foglio elettronico.

Tuttavia, per una maggiore chiarezza nella programmazione e una rappresentazione più precisa del codice, è importante specificare, oltre al nome della variabile, anche il tipo con la clausola **As** (come). Per esempio:

```
Dim V1 As Integer
```

per dichiarare una variabile *V1* di tipo numerico intero.

Tipi di dato

Integer	Numeri interi a 2 byte nell'intervallo da -32.768 a 32.767.
Long	Numeri interi di 4 byte, cioè numeri interi compresi tra -2.147.483.648 e 2.147.483.647.
Single	Numeri reali a virgola mobile e precisione singola a 32 bit (2 byte), compresi tra -3,402823E38 e -1,401298E-45 per valori negativi e tra 1,401298E-45 e 3,402823E38 per valori positivi.
Double	Numeri a virgola mobile e doppia precisione a 64 bit (8 byte) compresi tra -1,79769313486232E308 e -4,94065645841247E-324 per i valori negativi, tra 4,94065645841247E-324 e 1,797693134862325E308 per i valori positivi.
Date	Date e orari come numeri reali.
String	Dati composti da una sequenza di caratteri
Boolean	Dati con solo due valori possibili, <i>True</i> (-1) o <i>False</i> (0).

È possibile dichiarare diverse variabili in una singola istruzione:

```
Dim Contatore As Integer, AreaCerchio As Double, Nome As String  
Dim Trovato As Boolean
```

Le costanti sono dichiarate con la parola chiave **Const** in questo modo:

```
Const Max = 30
```

Si può, con più precisione, specificare anche il tipo di dato:

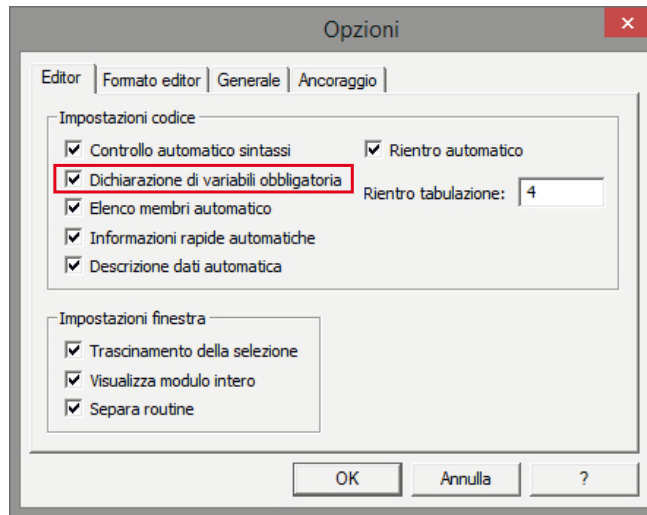
```
Const Max As Integer = 30
```

Dichiarazione esplicita delle variabili

Anche se il codice funziona correttamente senza dichiarare le variabili in modo esplicito con l'istruzione *Dim*, per programmare in modo ordinato è comunque opportuno e utile dichiarare le variabili in modo esplicito e con un tipo di dati specifico. La dichiarazione esplicita consente infatti di ridurre gli errori per conflitti di nome e gli errori di ortografia.

Per evitare che le variabili vengano dichiarate in modo implicito, si deve scrivere l'istruzione **Option Explicit** prima dei sottoprogrammi nella finestra del codice. Questa istruzione forza la dichiarazione esplicita di tutte le variabili nelle subroutine. Quando viene utilizzata l'istruzione *Option Explicit* e viene incontrato un nome di variabile non dichiarata o digitato in modo non corretto, sarà generato un errore di compilazione (*Variabile non definita*).

Nella finestra di Visual Basic è disponibile un'opzione che consente di includere automaticamente l'istruzione *Option Explicit*: dal menu **Strumenti**, scegliere **Opzioni** e nella scheda **Editor** mettere il segno di spunta in corrispondenza della scelta **Dichiarazione di variabili obbligatoria**.

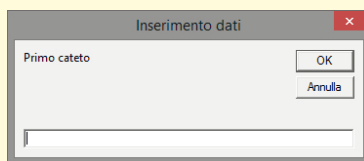


Input e output

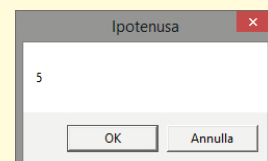
Le istruzioni e le funzioni del linguaggio Visual Basic consentono l'input dei dati da tastiera e la visualizzazione dei dati attraverso le finestre di dialogo tipiche del sistema operativo Windows. Il seguente sottoprogramma risolve il problema del calcolo dell'ipotenusa di un triangolo rettangolo, dopo aver acquisito le misure dei cateti.

```
Private Sub CommandButton1_Click()
' dichiarazione delle variabili
Dim c1, c2, ipot As Single
' acquisizione delle misure dei cateti
c1 = InputBox("Primo cateto", "Inserimento dati")
c2 = InputBox("Secondo cateto", "Inserimento dati")
' calcolo dell'ipotenusa
ipot = Sqr(c1 ^ 2 + c2 ^ 2)
' visualizzazione del risultato
MsgBox ipot, vbOKCancel, "Ipotenusa"
End Sub
```

InputBox è la funzione predefinita che acquisisce un dato dall'utente e lo assegna alla variabile alla sinistra del segno =. I due parametri della funzione, indicati tra parentesi, rappresentano rispettivamente il messaggio inviato all'utente per sollecitare l'inserimento del dato (*prompt*) e il testo che compare nella barra del titolo.



MsgBox è la funzione predefinita che visualizza un risultato o un messaggio in una finestra di dialogo: il primo parametro indica la variabile da visualizzare, il secondo **vbOKCancel** specifica i tipi di pulsanti contenuti nella finestra di dialogo (*OK* e *Annulla*), il terzo indica il testo che compare nella barra del titolo.



Con queste funzioni si realizza pienamente l'interazione con l'utente attraverso l'**interfaccia grafica** e non si utilizzano le celle del foglio elettronico, ma solo oggetti grafici: pulsante di comando e finestre di dialogo.

Riferimenti di cella

Utilizzando il metodo **Range** si fa riferimento a una cella o a un intervallo di celle secondo le modalità utilizzate normalmente nel foglio elettronico, cioè, per esempio, "A1" (riferimento di cella tra virgolette) per indicare la cella che si trova nella colonna A e nella riga 1.

Range("B:B")	Colonna B
Range("1:1")	Riga 1
Range("A:D")	Colonne comprese tra A e D
Range("1:10")	Righe comprese tra 1 e 10
Range("1:1,7:7,12:12")	Righe 1, 7 e 12
Range("A:A,C:C,G:G")	Colonne A, C e G
Range("A7")	Cella A7
Range("A1:C5")	Celle comprese tra A1 e C5
Range("A1:C5,G1:H5")	Selezione multipla di celle

In alternativa, con la proprietà **Cells** è possibile utilizzare i numeri di indice per il riferimento alle celle, cioè possiamo riferirci a una singola cella identificandola con una coppia di indici che sono rispettivamente il numero di riga e il numero di colonna della cella.

Per esempio:

```
Cells(3, 2)
```

indica la cella che si trova sulla riga 3 e sulla colonna B (colonna 2).

Gli indici sono ordinati, prima l'indice di riga e poi l'indice di colonna separati da virgola.

In sostanza *Cells(3,2)* corrisponde alla notazione *Range("B3")*.

La proprietà *Cells* risulta particolarmente efficace per l'esecuzione di cicli all'interno di un intervallo di celle, perché è possibile sostituire il numero di indice con una variabile.

Per gestire i riferimenti a intere righe o colonne, è possibile utilizzare rispettivamente la proprietà **Rows** o la proprietà **Columns**.

Rows(1)	Riga 1
Columns(1)	Colonna 1
Columns("A")	Colonna 1
Rows	Tutte le righe del foglio di lavoro
Columns	Tutte le colonne del foglio di lavoro

Per esempio, il sottoprogramma:

```
Sub Grassetto()  
Rows(1).Font.Bold = True  
End Sub
```

evidenzia in grassetto (**Bold**) tutte le celle della prima riga del foglio elettronico, attraverso l'assegnazione del valore vero (*True*) alla proprietà *Bold* del tipo di carattere (*Font*) assegnato alle celle della riga 1.

La struttura di selezione

La struttura di **selezione** viene rappresentata secondo lo schema:

```
IF condizione THEN  
    istruzione1  
ELSE  
    istruzione2  
END IF
```


Se la condizione è vera, viene eseguita l'*istruzione1*, altrimenti viene eseguita l'*istruzione2*.
Istruzione1 e *istruzione2* possono indicare, come accade nella maggior parte dei casi, non una sola istruzione, ma un gruppo di istruzioni.

La condizione è un'espressione booleana di cui viene valutata la verità: vengono quindi utilizzati i segni del confronto: <, >, =, >=, <=, <> (*diverso*), e gli operatori booleani AND, NOT, OR per costruire espressioni logiche combinando tra loro più condizioni.

Le strutture di ripetizione

La **ripetizione con contatore** o **iterazione enumerativa** si rappresenta con la struttura:

```
FOR contatore = iniziale TO finale  
    istruzioni  
NEXT
```

Le istruzioni racchiuse tra *For* e *Next* sono ripetute tante volte quante occorrono per portare il *contatore* dal valore *iniziale* al valore *finale* incrementandolo di 1 ad ogni ripetizione.

Ripetizione per falso: nella **ripetizione per falso** (o **ripetizione post-condizionale**) il controllo viene fatto in coda, dopo aver eseguito le istruzioni del ciclo.

```
DO  
    istruzioni  
LOOP UNTIL condizione
```

La ripetizione termina quando la condizione scritta vicino a *Until* diventa vera.
La condizione può contenere anche gli operatori booleani AND, OR, NOT.

La **ripetizione per vero** (o **ripetizione pre-condizionale**, con controllo in testa) è rappresentata con:

```
DO WHILE condizione  
    istruzioni  
LOOP
```

La ripetizione viene eseguita mentre la condizione scritta vicino a *While* si mantiene vera.

Ripetizione *for each*

```
FOR EACH elemento IN gruppo  
    istruzioni  
NEXT
```

dove *gruppo* indica un insieme di oggetti, per esempio un insieme di celle alle quali è stato assegnato un nome, ed *elemento* rappresenta un singolo oggetto dell'insieme.
L'elemento deve essere dichiarato di tipo *Variant*.

Variant in Visual Basic è il tipo di dati in cui vengono trasformate tutte le variabili che non siano state dichiarate esplicitamente, come tipo diverso, attraverso l'istruzione *Dim*.