

Programmazione in Excel con il linguaggio Visual Basic

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**, già utilizzata in precedenza per la registrazione delle macro.



In particolare nel seguito useremo i pulsanti:



Visual Basic nel gruppo **Codice**, per scrivere il testo sorgente della procedura.



Inserisci nel gruppo **Controlli** per mettere sul foglio elettronico gli oggetti grafici (controlli).



Modalità progettazione, nel gruppo **Controlli**, per attivare la creazione del codice oppure per uscire dalla progettazione e passare alla prova di esecuzione delle procedure.



Proprietà, nel gruppo **Controlli**, per impostare le caratteristiche degli oggetti grafici.

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

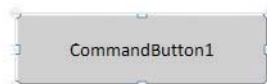
- fare clic sul pulsante **Modalità progettazione**;
- fare clic sul pulsante **Inserisci** e, nell'elenco **Controlli ActiveX**, scegliere un controllo grafico, per esempio un **Pulsante di comando** (passando con il mouse sopra gli oggetti si può evidenziare il nome dell'oggetto).




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.

- disegnare l'oggetto sul foglio Excel trascinando il mouse;

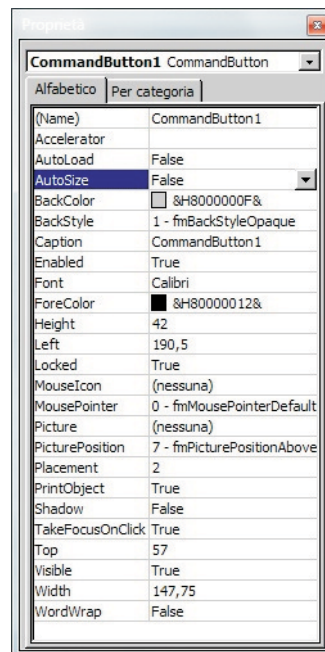
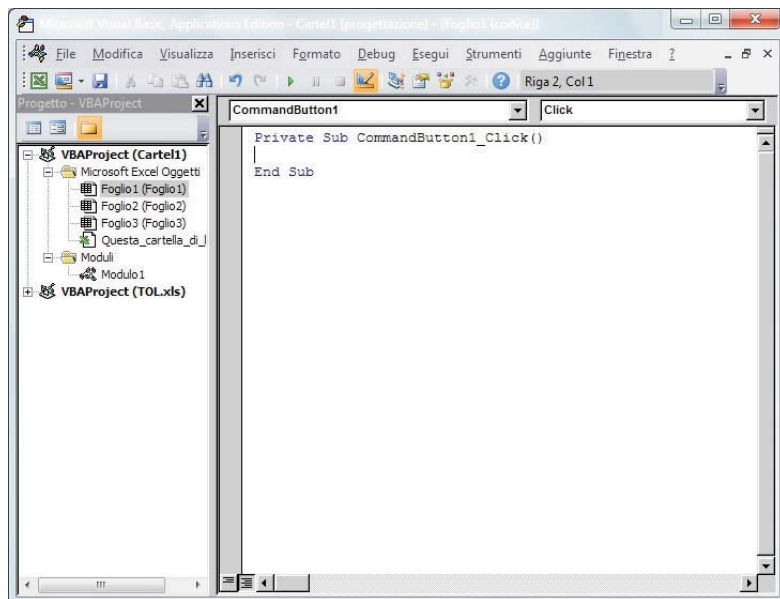


- fare doppio clic sull'oggetto per attivare l'editor del codice;
- nella **finestra del codice** che si apre scrivere le istruzioni in linguaggio Visual Basic;
- tornare al foglio Excel facendo clic sulla prima icona a sinistra nella barra degli strumenti della finestra del codice (con il logo di Excel ) oppure facendo clic sul pulsante del foglio Excel nella barra delle applicazioni in basso sullo schermo;
- uscire dalla modalità progettazione facendo clic sul pulsante **Modalità progettazione**;
- provare l'esecuzione del codice facendo clic sull'oggetto grafico.

La scorciatoia da tastiera per passare velocemente dal foglio di Excel alla finestra del codice Visual Basic, e viceversa, è la combinazione dei tasti **Alt + F11**.

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 visualizzata facendo clic su di esso con il pulsante destro del mouse e scegliendo **Proprietà** dal menu di scelta rapida. Oppure si può fare clic sul pulsante **Proprietà** nel gruppo **Controlli** della scheda **Sviluppo**.

Facendo doppio clic su un oggetto, in modalità progettazione, si attiva la **finestra del codice**, nella quale si scrivono le istruzioni 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 cosa deve fare il computer come risposta alle azioni dell'utente. I possibili eventi vengono visualizzati facendo clic sulla casella a destra nella finestra del codice (casella *Routine*), accanto alla casella degli oggetti (*Oggetto*).

Le istruzioni sono raggruppate tra la frase **Sub** e la frase **End Sub**, per indicare l'inizio e la fine del sottoprogramma (*Subroutine*) associato all'evento. L'instestazione del sottoprogramma contiene anche il nome dell'oggetto grafico (nell'esempio della figura, *CommandButton1*) e l'evento da gestire (nella figura, *Click*).

Oltre alla combinazione di tasti **Alt+F11**, per passare dal foglio elettronico alla finestra del codice è possibile anche usare il pulsante **Visual Basic** nel gruppo **Codice** della scheda **Sviluppo**.

Si può anche fare clic con il tasto destro del mouse sopra l'oggetto grafico e selezionare **Visualizza codice** dal menu di scelta rapida. Dal codice si può poi tornare all'oggetto con la combinazione di tasti **Maiusc + F7**.

Istruzioni in sequenza

Il problema seguente presenta un esempio di codifica in Visual Basic della **sequenza**, cioè la struttura dell'algoritmo nella quale le istruzioni sono indicate una di seguito all'altra secondo l'ordine con il quale devono essere eseguite dal computer.

Progetto 1

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

- *Dati di input:* le misure dei due cateti, $c1$, $c2$
- *Dati di output:* la misura dell'ipotenusa $ipot$.
- *Risoluzione:* dopo aver acquisito i valori di $c1$ e $c2$, si calcola l'ipotenusa $ipot$ con la formula $ipot = \sqrt{c1^2 + c2^2}$

algoritmo TriangoloRettangolo

variabili

dichiara $c1$, $c2$,
 $ipot$ come numeri reali

inizio

immetti $c1$
immetti $c2$
assegna $ipot = \sqrt{c1^2 + c2^2}$
scrivi $ipot$

fine

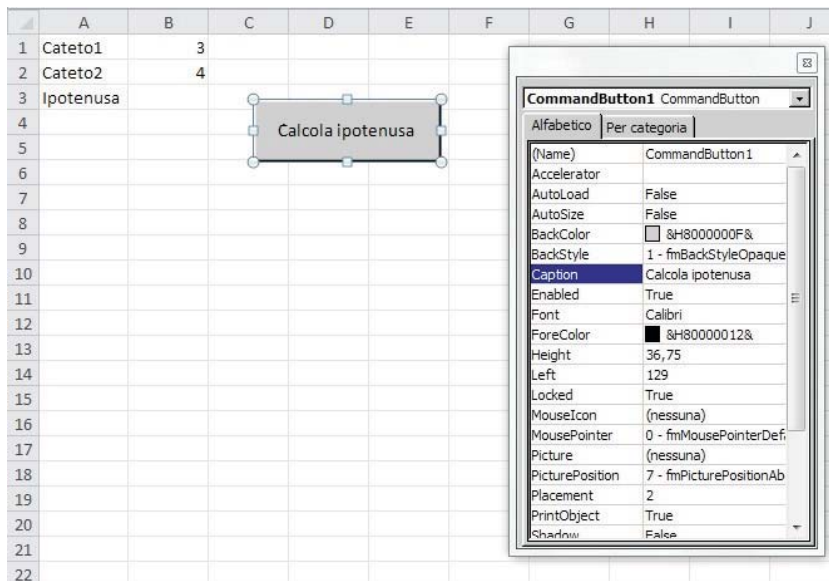
riga di intestazione

sezione dichiarativa

sezione esecutiva

Interfaccia utente

Definiamo la disposizione dei dati e degli oggetti grafici sul foglio elettronico.



Disegniamo un pulsante di comando (scegliendolo dalla casella dei **Controlli ActiveX**) per consentire all'utente di attivare il codice. Facendo clic con il pulsante destro del mouse e scegliendo **Proprietà** dal menu di scelta rapida, è possibile cambiare la proprietà **Caption**, che rappresenta il testo che compare sopra il pulsante di comando: scriviamo per esempio *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 grafico 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
```

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

Vediamo il significato delle istruzioni precedenti:

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. I tipi di dati utilizzabili in Visual Basic sono spiegati di seguito al progetto.

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

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.

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

Si osservi anche che 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**.

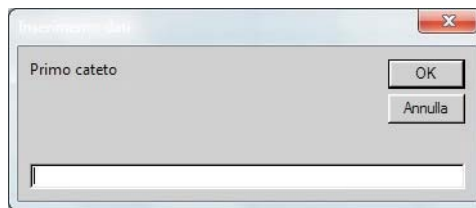
Facendo ora clic sul pulsante di comando con la scritta *Calcola ipotenusa*, il sottoprogramma viene eseguito e nella cella B3 compare il valore calcolato.

Il problema del calcolo dell'ipotenusa può anche essere risolto utilizzando le istruzioni e le funzioni del linguaggio Visual Basic che consentono l'input dei dati da tastiera e la visualizzazione dei dati attraverso le finestre di dialogo tipiche del sistema operativo Windows.

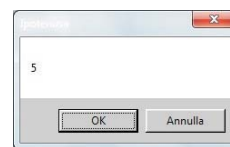
Riscriviamo il sottoprogramma in questo modo:

```
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
```

In esso **InputBox** indica 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 nella finestra di dialogo.



L'istruzione **MsgBox** indica la visualizzazione di un risultato o di 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 nella finestra di dialogo.



Il sottoprogramma viene eseguito, come nella versione precedente, facendo clic sul pulsante *Calcola ipotenusa* dopo aver disattivato la *Modalità progettazione*.

In questa seconda versione 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.

Dichiarazione di costanti e variabili

Abbiamo visto nell'esempio precedente che 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.

I tipi di dato più usati sono:

Integer, per variabili memorizzate come numeri interi a 2 byte nell'intervallo da -32.768 a 32.767.

Long, per interi di 4 byte, cioè numeri interi compresi tra -2.147.483.648 e 2.147.483.647.

Single, per 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, per 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, per memorizzare date e orari come numeri reali.

String, per dati composti da una sequenza di caratteri

Boolean, per dati con solo due valori possibili, *True* (-1) o *False* (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
```

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, è possibile posizionare 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, verrà generato un errore di compilazione (*Variabile non definita*) quando si esegue il programma.

Nell'ambiente di programmazione 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**.

I riferimenti di cella

Utilizzando il metodo **Range**, visto nel Progetto 1, è possibile riferirsi 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.

Per esempio:

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 una 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, come vedremo negli esempi seguenti.

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

Per esempio:

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 in Visual Basic 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.

Si può osservare che la struttura IF ... THEN ... ELSE ... traduce nel linguaggio Visual Basic la funzione **SE** del foglio elettronico Excel.

Progetto 2

Dati due numeri, diversi tra loro, disporli in ordine crescente.

- *Dati di input*: due numeri indicati con Primo e Secondo
- *Dati di output*: i due numeri ordinati indicati con Minore e Maggiore.
- *Risoluzione*: i due numeri acquisiti vengono confrontati; se il primo è minore del secondo si assegna alla variabile Minore il primo numero e alla variabile Maggiore il secondo; altrimenti le assegnazioni sono scambiate.

algoritmo Ordina

riga di intestazione

variabili

sezione dichiarativa

dichiara Primo, Secondo,
Minore, Maggiore come numeri interi

Inizio

sezione esecutiva

immetti Primo
immetti Secondo
se Primo < Secondo
 allora
 assegna Minore = Primo
 assegna Maggiore = Secondo
 altrimenti
 assegna Minore = Secondo
 assegna Maggiore = Primo

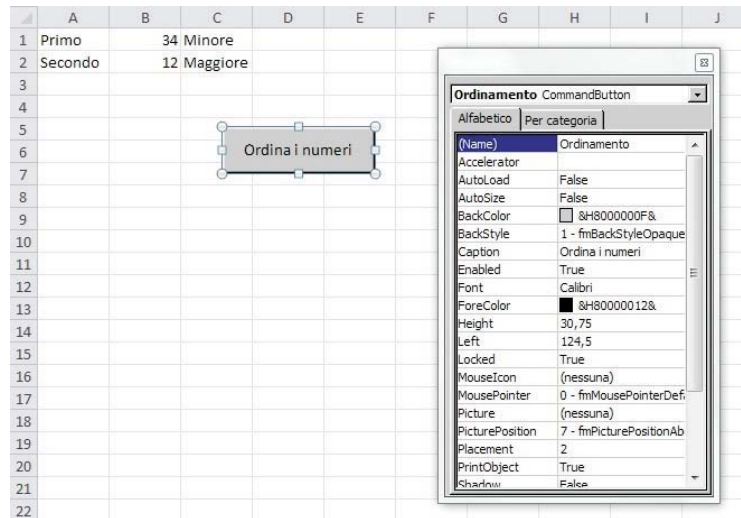
fine se

fine

Interfaccia utente

Disponiamo i numeri nel foglio elettronico e inseriamo un pulsante di comando per attivare il controllo dei numeri. Nella colonna B l'utente inserisce i due numeri, il programma dispone i due numeri ordinati in modo crescente nella colonna D.

Alla proprietà **Caption** del pulsante di controllo è assegnato il valore *Ordina i numeri*. È anche opportuno assegnare un nome al controllo, attraverso la proprietà **Name**, per meglio identificarlo: questo risulta particolarmente utile quando i controlli sono più di uno. Il nome assegnato al controllo viene poi automaticamente utilizzato nel nome del sottoprogramma associato al controllo. Possiamo assegnare il valore *Ordinamento* alla proprietà *Name*.



Codice

In questo sottoprogramma useremo la notazione **Cells** per il riferimento alle celle, come spiegato nel precedente paragrafo.

```
Option Explicit

Private Sub Ordinamento_Click()
' dichiarazione delle variabili
Dim Primo, Secondo As Integer
Dim Minore, Maggiore As Integer
' assegnazione dei valori
Primo = Cells(1, 2)
Secondo = Cells(2, 2)
' controllo dei numeri
If Primo > Secondo Then
    Minore = Secondo
    Maggiore = Primo
Else
    Minore = Primo
    Maggiore = Secondo
End If
' numeri ordinati
Cells(1, 4) = Minore
Cells(2, 4) = Maggiore
End Sub
```

Le strutture di ripetizione per falso e per vero

La struttura di ripetizione può essere rappresentata in modi diversi. La prima forma riguarda la **ripetizione per falso** o **ripetizione post-condizionale**, cioè la ripetizione nella quale il controllo viene fatto in coda, dopo aver eseguito le istruzioni del ciclo.

Questa struttura è tradotta in Visual Basic con le istruzioni **DO ... LOOP UNTIL**.

La sintassi generale è la seguente:

```
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.

Progetto 3

Generare un numero casuale superiore a 40.

- *Dato di output*: numero generato.
- *Risoluzione*: attraverso la funzione predefinita per la generazione dei numeri casuali (funzione per la generazione di numeri *random* o funzione *randomizzatrice*), il programma genera un numero casuale compreso tra 1 e 100 e lo scrive in una cella del foglio di calcolo; l'operazione viene ripetuta finché il numero generato diventa maggiore di 40.

algoritmo NumeroCasuale

```
variabili
    dichiara Numero come numero intero
inizio
    esegui
        genera Numero
    ripeti finché Numero > 40
    scrivi Numero
fine
```

riga di intestazione

sezione dichiarativa

sezione esecutiva

Si osservi che la **generazione di numeri casuali** compresi in un intervallo è uno strumento importante nella programmazione per testare le funzionalità di un programma e la correttezza del codice, ma anche per disporre di dati numerici di prova senza la necessità di inserimenti da tastiera.

Interfaccia utente

Nel foglio viene predisposta la cella per contenere il numero generato e un pulsante di comando per avviare la procedura di calcolo.

	A	B	C	D	E	F	G
1	numero generato	76					
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							

Genera il numero casuale

Codice

Il programma utilizza la funzione predefinita **Rnd** (*random*) del linguaggio Visual Basic, che genera numeri reali casuali compresi tra 0 e 1 (1 escluso). Moltiplicando la funzione *Rnd* per 100, si ottengono quindi numeri casuali compresi tra 0 e 99.

Per ottenere numeri casuali compresi tra 1 e 100 si aggiunge 1:

$Rnd * 100 + 1$.

Per ottenere poi numeri interi si applica alla formula la funzione predefinita **Int** del linguaggio Visual Basic che restituisce la parte intera di un numero reale:

$Int(Rnd * 100 + 1)$

La ripetizione, che genera i numeri e li scrive nella cella B1, si arresta quando il numero generato diventa superiore a 40.

```
Private Sub GeneraNumero_Click()
' dichiarazione delle variabili
Dim numero As Integer
' ripetizione per falso
Do
' generazione di un numero casuale
' compreso tra 1 e 100
numero = Int(Rnd * 100 + 1)
Loop Until numero > 40
Cells(1, 2) = numero
End Sub
```

Lo stesso problema può essere risolto in forma equivalente usando la **ripetizione per vero** o **ripetizione con controllo in testa**, che in Visual Basic è rappresentata con la struttura **DO WHILE ... LOOP**.

La sintassi generale della struttura è la seguente:

```
DO WHILE condizione
    istruzioni
LOOP
```

La ripetizione viene eseguita mentre la condizione scritta vicino a *While* si mantiene vera. All'inizio del programma alla variabile *numero* è assegnato il valore 0 per poter forzare l'esecuzione della ripetizione almeno una volta, effettuando il primo controllo della condizione alla partenza della ripetizione.

```
Private Sub GeneraNumero_Click()
' dichiarazione delle variabili
Dim numero As Integer
numero = 0
' ripetizione per vero
Do While numero <= 40
' generazione di un numero casuale
' compreso tra 1 e 100
numero = Int(Rnd * 100 + 1)
Loop
Cells(1, 2) = numero
End Sub
```

Si noti che la condizione scritta vicino a *While* è l'opposto (cioè la negazione booleana) della condizione scritta vicino a *Until* nel programma precedente.

La struttura di ripetizione con contatore

La **ripetizione con contatore** o **iterazione enumerativa** si rappresenta con la struttura **FOR ... NEXT**.

La sintassi generale della struttura è la seguente:

```
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.

Progetto 4

Visualizzare la tavola pitagorica in un foglio Excel.

- *Dati di output*: i numeri della tavola pitagorica da 1 a 9.
- *Risoluzione*: I valori da inserire nella tavola pitagorica si ottengono moltiplicando tra loro il numero di riga e il numero di colonna; per ciascuna riga, l'indice varia tra 1 e 9 per le colonne. Si utilizzano quindi due ripetizioni: una esterna per le righe e una interna per le colonne.

algoritmo TavolaPitagorica

variabili

dichiara r, c come numero intero

inizio

- - - ripetizione sulle righe - - -

per r = 1 fino a 9

- - - per ciascuna riga ripetizione sulle colonne - - -

per c = 1 fino a 9

assegna elemento della tavola = r * c

ripeti

ripeti

fine

riga di intestazione

sezione dichiarativa

sezione esecutiva

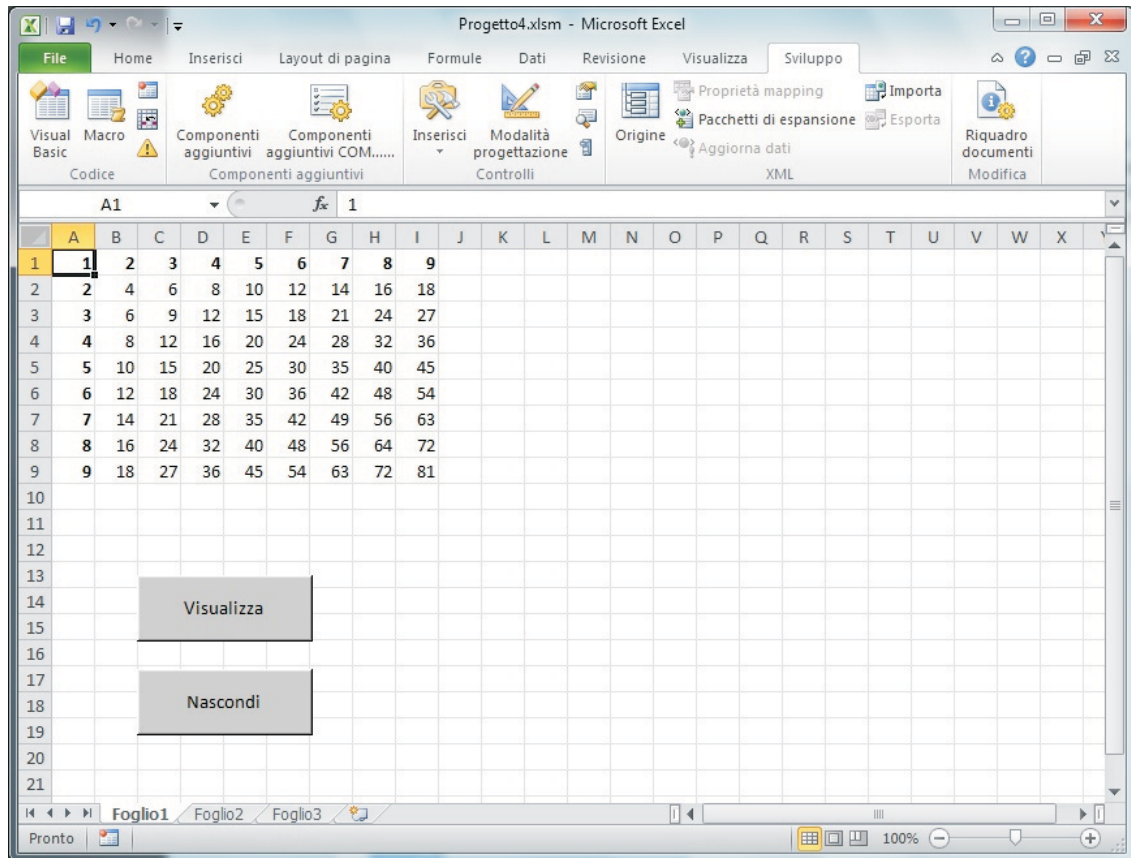
Interfaccia utente

Nel foglio di lavoro vengono inseriti due pulsanti di comando per visualizzare o per nascondere i numeri della tavola pitagorica da 1 a 9.

Ai due pulsanti sono assegnate le seguenti proprietà:

primo pulsante: proprietà *Name* e proprietà *Caption* = *Visualizza*;

secondo pulsante: proprietà *Name* e proprietà *Caption* = *Nascondi*.



Codice

Il primo sottoprogramma (*Visualizza*) scrive i valori calcolati nelle celle, inoltre evidenzia in grassetto la prima riga e la prima colonna della tavola, e riduce alla metà la larghezza delle colonne. Quando si decide di nascondere i numeri con il secondo sottoprogramma (*Nascondi*), i numeri vengono tolti dal foglio e la larghezza delle colonne viene riportata alla misura standard.

```
Private Sub Visualizza_Click()  
    ' dichiarazione delle variabili  
    Dim r, c As Integer  
    ' assegnazione dei valori alle celle  
    For r = 1 To 9  
        For c = 1 To 9  
            Cells(r, c) = r * c  
        Next c  
    End For  
End Sub
```

```

Next r
' grassetto alla prima riga e cambiamento di larghezza
Rows(1).Font.Bold = True
Columns.ColumnWidth = 4
' grassetto alla prima colonna
Columns("A").Font.Bold = True
End Sub

Private Sub Nascondi_Click()
' dichiarazione delle variabili
Dim r, c As Integer
' toglie i numeri dalle celle
For r = 1 To 9
    For c = 1 To 9
        Cells(r, c) = ""
    Next c
Next r
' cambiamento di larghezza
Columns.ColumnWidth = 8.43
End Sub

```

Interfaccia grafica

Nei progetti precedenti abbiamo utilizzato le righe e le colonne del foglio elettronico per memorizzare i dati di input e i risultati dell'elaborazione, oltre a pulsanti di comando (oggetti **CommandButton**) per attivare l'avvio delle procedure. In alcuni casi abbiamo anche utilizzato finestre di dialogo per acquisire i dati (**InputBox**) o per visualizzare i risultati (**MsgBox**).

Con il linguaggio Visual Basic è possibile anche costruire sottoprogrammi senza utilizzare le celle del foglio elettronico, ma solo oggetti grafici, tipici del sistema operativo Windows, scegliendoli dalla casella dei **Controlli ActiveX**: in particolare etichette, caselle di testo e pulsanti di comando, che sono gli elementi grafici più usati per costruire le interfacce grafiche per l'utente (**GUI**, *Graphical User Interface*).

In questi casi è opportuno nascondere la griglia del foglio elettronico in questo modo: dal menu **File** scegliere **Opzioni** e poi **Impostazioni avanzate**: nella sezione **Opzioni di visualizzazione per il foglio di calcolo**, togliere il segno di spunta alla voce **Mostra Griglia**.

In alternativa, nella scheda **Layout di pagina**, gruppo **Opzioni del foglio**, togliere il segno di spunta all'opzione **Visualizza di Griglia**.

Il seguente progetto tratta un problema volutamente semplice, per consentire di fissare l'attenzione sugli strumenti che possono essere utilizzati per costruire l'interfaccia grafica.

Progetto 5

Calcolare l'area di un trapezio.

- *Dati di input:*
misure di base maggiore, base minore e altezza
- *Dati di output:* area del trapezio.
- *Risoluzione:* dopo aver acquisito le misure di base maggiore B , base minore b e altezza h , l'area si calcola con la formula

$$\text{Area} = (B + b) * h / 2$$

algoritmo Area trapezio

variabili

dichiara BaseMaggiore, BaseMinore, Altezza,
Area come numeri reali

inizio

immetti BaseMaggiore, BaseMinore, Altezza
assegna Area = (BaseMaggiore + BaseMinore) * Altezza / 2
scrivi Area

fine

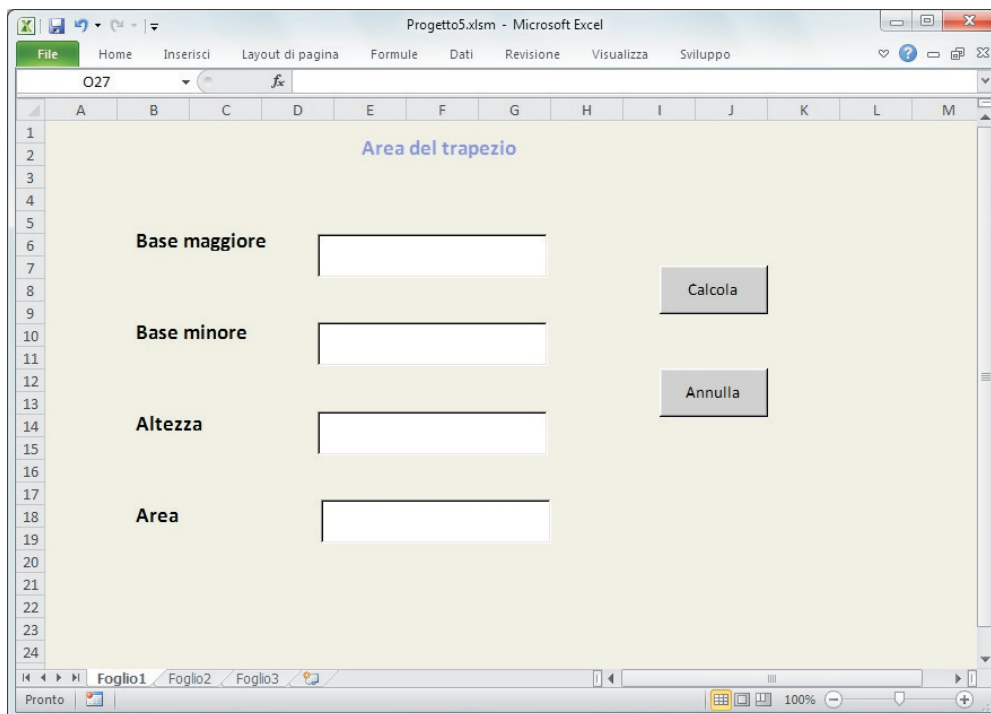
[riga di intestazione](#)

[sezione dichiarativa](#)

[sezione esecutiva](#)

Interfaccia utente

Definiamo la disposizione dei dati e degli oggetti grafici sul piano di lavoro. Nel foglio è stata tolta la griglia secondo la modalità descritta in precedenza ed è stato assegnato un colore di sfondo alle celle: dopo aver selezionato l'intero foglio (clic sul pulsante posto all'incrocio tra la riga di intestazione delle colonne e la colonna di intestazione delle righe), nella scheda **Home**, gruppo **Carattere**, selezionare il colore dal pulsante **Colore riempimento**.



Sono stati poi inseriti alcuni elementi grafici scegliendoli dalla casella dei *Controlli ActiveX*:

- un' **Etichetta** (in inglese *Label*) per il titolo del programma (*Area del trapezio*);
- quattro **Etichette** per le descrizioni delle caselle per i dati (*Base maggiore, Base minore, Altezza, Area*).

Per tutte queste etichette è stata cambiata la proprietà **BackStyle** (stile dello sfondo) impostandolo come **Transparent** (trasparente) in modo da ottenere uno sfondo uguale allo sfondo dell'intero foglio; inoltre è stato modificato il font in grassetto con la proprietà **Font** e il colore del titolo con la proprietà **ForeColor**; la descrizione dell'etichetta è impostata con la proprietà **Caption**.

Le etichette vanno disposte in modo ordinato e allineato nel piano di lavoro: per far questo occorre selezionare con il mouse gli oggetti tenendo premuto il tasto **Ctrl** e poi, nella scheda **Layout di pagina**, gruppo **Disponi**, fare clic sul pulsante **Allinea**: applicare due opzioni, prima **Allinea al centro verticalmente** e poi **Distribuisci verticalmente**.

- quattro **Caselle di testo** (in inglese *TextBox*) per i dati del problema: per esse è stata impostata la proprietà **Name**, rispettivamente in *TBaseMaggiore, TBaseMinore, TAltezza, TArea*, che sono i nomi che verranno usati nel codice.

Anche le caselle di testo devono essere ben allineate e distanziate con la stessa modalità indicata per le etichette.

- due **Pulsanti di comando** (in inglese *CommandButton*), per i quali sono state impostate le proprietà **Name**, per identificarli con più chiarezza nel codice, e **Caption** per la descrizione che deve comparire sul pulsante.

Il primo pulsante *Calcola* attiva la procedura di calcolo dell'area, il pulsante *Annulla* cancella i dati contenuti nelle caselle di testo, per consentire un nuovo calcolo con dati diversi.

Per completare la disposizione degli oggetti, è opportuno allineare anche i pulsanti di comando in modo analogo alle etichette e alle caselle di testo.

Codice

I due sottoprogrammi seguenti rappresentano le istruzioni da eseguire, associate all'evento *Click* per ciascuno dei due pulsanti di comando:

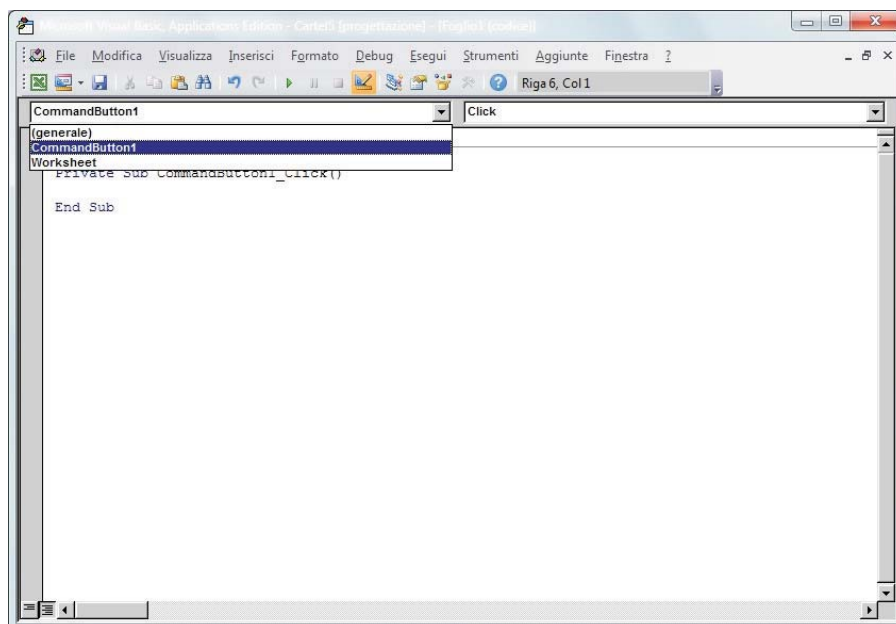
- il primo, *Annulla*, cancella il contenuto delle caselle di testo, offrendo la possibilità di inserire nuovi dati,
- il secondo, *Calcola*, acquisisce i dati contenuti nelle prime tre caselle di testo, calcola l'area e ne scrive il valore nella terza casella di testo.

```
Private Sub Annulla_Click()  
    TBaseMaggiore = ""  
    TBaseMinore = ""  
    TAltezza = ""  
    TArea = ""  
End Sub  
  
Private Sub Calcola_Click()  
    TArea = (Val(TBaseMaggiore) + Val(TBaseMinore)) * TAltezza / 2  
End Sub
```

Si noti l'uso della funzione predefinita **Val** del linguaggio Visual Basic, che restituisce il valore numerico di una stringa contenente cifre. Il contenuto di una casella di testo (*TextBox*), infatti, è una stringa di caratteri: se non si usasse la funzione *Val* il risultato dell'addizione produrrebbe semplicemente un accodamento dei caratteri della seconda stringa ai caratteri della prima stringa (somma di stringhe).

La finestra del codice

La **finestra del codice** viene usata per inserire, visualizzare e modificare righe di codice. Si possono aprire anche tante finestre di codice, in modo da vedere più finestre contemporaneamente, e anche per poter fare operazioni di copia e incolla tra una procedura e l'altra.



La finestra comprende due liste:

- la **lista degli Oggetti** che contiene i controlli in essa presenti (a sinistra);
- la **lista delle Routine** che contiene la lista di tutte le azioni che possono essere provocate dall'utente, associate da Visual Basic ai controlli presenti nella lista degli oggetti (a destra).

Selezionando un'azione, nella parte alta della finestra, viene presentata la procedura associata a quell'azione o uno schema generale di procedura che può essere completato.

Se nella lista degli oggetti è visualizzata la parola **Generale**, nella lista delle routine vengono presentate le procedure e le dichiarazioni di carattere generale, per esempio la clausola *Option Explicit* per la dichiarazione esplicita delle variabili.

Il codice Visual Basic viene inserito ed editato usando la finestra di codice; essa offre al programmatore due supporti importanti:

- la **formattazione automatica**, cioè il testo del codice viene automaticamente formattato man mano che viene inserito da tastiera, rendendo maiuscole le prime lettere delle parole-chiave, regolarizzando gli spazi, usando diversi colori per il testo e per le parole-chiave.

In particolare:

- il testo del codice è scritto normalmente in colore nero,
 - le parole-chiave del linguaggio sono indicate con il colore azzurro,
 - le frasi di commento sono scritte in verde,
 - le righe di codice contenenti errori sono evidenziate con il colore rosso.
- il **controllo sintattico**, che effettua il controllo sul codice durante l'attività di editing stessa, segnalando gli errori di sintassi con un messaggio.