Nozioni Generali sul VBA

In Excel, una serie di codici VBA sono chiamati *macro* o *procedure* e il VBA è un linguaggio di programmazione utilizzato per lavorare con Microsoft Excel, ma anche con altre applicazioni Microsoft Office come Word, Access, PowerPoint, etc. In sintesi si tratta di un linguaggio di programmazione che interagisce con Excel e viene usato per programmare e automatizzare le attività. La programmazione VBA è utilizzata per ottenere una migliore funzionalità di calcolo, per automatizzare le operazioni ripetitive e per integrare Excel con altre applicazioni di Office come Microsoft Access. Le istruzioni date a Excel sono sotto forma di codice, chiamati macro o procedure e il codice è sviluppato in Visual Basic Editor (VBE), che è l'ambiente di sviluppo VBA.

Excel VBA oggetti, Proprietà e metodi

Un oggetto è una cosa che contiene i dati e ha ***proprietà*** e ***metodi***. Le proprietà sono le caratteristiche o gli attributi che descrivono l'oggetto, come il nome, il colore, la dimensione, o definiscono il comportamento di un oggetto, se è visibile o abilitato. I dati o le informazioni di un oggetto possono essere raggiunti con proprietà o metodi, dove il metodo è un'azione eseguita da un oggetto tramite un codice VBA che farà sì che l'oggetto esegua un'azione.

Per meglio comprendere si pensi ad un oggetto come a una casa o un'auto, le proprietà di una vettura comprendono il suo colore o le dimensione che la descrivono, inoltre una vettura può eseguire azioni di movimento o accelerazione che sono i suoi metodi. Esempi di oggetti in Excel sono la cartella di lavoro, il foglio di lavoro, un pulsante di comando, i font, etc. Un oggetto ***Range*** ha un "valore", che è una delle sue proprietà e "*Select*" uno dei suoi metodi. Allo stesso modo un foglio di lavoro ha, tra l'altro, una proprietà "*Name*", un metodo "*Elimina*", e un metodo "*Copia*" avendo argomenti che contengono informazioni per quanto riguarda il foglio da copiare.

Il modello a oggetti ***Application*** (Excel) si riferisce e contiene i suoi oggetti di programmazione che sono legati gli uni agli altri in una gerarchia e l'intera applicazione Excel è rappresentata dall'oggetto Application che è in cima alla gerarchia di oggetti di Excel e spostandosi verso il basso è possibile accedere all'oggetto cartella di lavoro, ai fogli di lavoro, al Range (Celle) etc.

Procedure di evento in VBA

Gli oggetti hanno anche procedure di evento ad essi connessi, che sono azioni eseguite, o innescate da altri eventi, che eseguono un codice VBA, oppure da macro. Una routine evento (es. un codice VBA) viene attivato quando si verifica un evento come l'apertura, la chiusura, l'attivazione o disattivazione della cartella di lavoro, la selezione di una cella o cambiando la selezione delle celle in un foglio di lavoro, fare un cambiamento nel contenuto di un foglio di lavoro, la selezione o l'attivazione di un foglio di lavoro, e così via. Excel dispone di routine evento che sono procedure richiamate automaticamente quando un oggetto riconosce il verificarsi di un evento. Le procedure di evento sono collegate a oggetti come la cartella di lavoro, il foglio di lavoro, i grafici, le Form o i vari controlli.

Le procedure di evento sono attivate da un evento predefinito e vengono installate all'interno di Excel con un nome standard e predeterminato, come la procedura di modifica del foglio di lavoro viene installato con il foglio di lavoro e denominata "*Private Sub Worksheet_Change (ByVal Target As Range)*". Nella routine evento Worksheet Change, l'oggetto foglio di lavoro è associato all'evento Change, il che significa che con l'evento di modifica Worskheet, contenente un codice personalizzato la routine viene eseguita automaticamente quando si modifica il contenuto di una cella del foglio di lavoro.

Visual Basic Editor (VBE)

VBE è contenuto nella cartella di lavoro di Microsoft Excel, ed è un ambiente usato per scrivere, modificare ed eseguire il debug di codice VBA che si può avviare in Excel 2007 dai seguenti percorsi:

- Dalla scheda **Sviluppo - Visual Basic**
- Dalla scheda **Sviluppo - Visualizza codice**
- Fare clic col destro del mouse sulla scheda del nome foglio in basso, quindi fare clic su **Visualizza codice**
- Premere la combinazione di tasti **Alt + F11**

I componenti di Visual Basic Editor si riferiscono alla *Finestra del Codice*, *la Finestra Gestione progetti*, *la finestra Proprietà* e l'area di lavoro di programmazione (es. menu e barre degli strumenti, oggetti, la finestra Immediata e la finestra di controllo). La Finestra del codice è dove scrivere e modificare il codice e le procedure, e anche dove le macro vengono registrate, mentre esplora progetti visualizza l'elenco di tutti i progetti esistenti e fornisce una vista ad albero in cui è possibile comprimere, per nascondere, o espandere, per visualizzare, gli oggetti, i Form e i moduli contenuti in un progetto. Ogni progetto contiene una cartella oggetti, che è il *Microsoft Excel Objects*, una cartella Forms che contiene i Form, una cartella Moduli che contiene i moduli e la cartella moduli di classe che contiene le classi.

La cartella Objects è sempre presente e contiene un oggetto foglio per ogni foglio di lavoro esistente, e un oggetto il *ThisWorkbook* e ogni oggetto foglio ha un primo nome che appare prima delle due parentesi, che è il nome del foglio e un secondo nome che compare tra le parentesi che è il nome della scheda del foglio che appare nel foglio di lavoro di Excel. Per visualizzare la finestra di Esplora progetti, si deve cliccare su *Visualizza* sulla barra dei menu VBE e quindi selezionare *Esplora progetti*, o premere CTRL + R in VBE.

Moduli in Excel VBE

Le macro VBA (cioè i codici o le procedure) devono risiedere nei loro appositi moduli o Excel non sarà in grado di trovare ed eseguirli, l'oggetto Modulo viene utilizzato per le procedure di evento incorporate in Excel e per creare i propri eventi che sono: Modulo, *ThisWorkbook*, moduli Sheet (fogli di lavoro e fogli grafici), moduli Form e moduli di classe, in generale il codice VBA nei confronti degli eventi non associati a un particolare oggetto (come la cartella di lavoro o un foglio) vengono inseriti in un modulo di codice standard, una pratica generalmente accettata è quella di piazzare le routine di evento nel modulo ThisWorkbook, nei moduli di fogli e UserForm, pur ponendo altro codice VBA in moduli standard. I moduli di codice standard sono anche indicati come moduli di codice o moduli, e ci possono essere vari moduli (Modulo1, Modulo2 etc.) in un progetto VBA, in cui ciascun modulo può essere utilizzato per coprire un certo aspetto del progetto. Per inserire un modulo si deve cliccare su *Inserisci* sulla barra dei menu VBE e quindi selezionare *Modulo*.

Procedure VBA

Una procedura VBA, indicata anche come una macro è definita come un insieme di codici che permettono di eseguire un'azione ed è generalmente di due tipi, un sub-procedimento (cioè sub-routine) o una funzione. Il terzo tipo di procedura è utilizzata per moduli di classe. Un progetto VBA può contenere più moduli, moduli di classe e le Form e ogni modulo contiene una o più procedure, sub-procedure o funzioni.

Creazione di un sub-procedimento

Un sub-procedimento inizia con la dichiarazione della parola chiave ***Sub***, seguita da un nome e poi seguita da una serie di parentesi e si conclude con una dichiarazione ***End Sub*** che può essere digitato, ma appare automaticamente dopo aver digitato il nome della procedura seguito dalla pressione del tasto Invio per andare alla riga successiva. Il codice VBA o le dichiarazioni sono inseriti in mezzo tra le due dichiarazioni

Eseguire una procedura

Ci sono molti modi per eseguire una macro, utilizzando la finestra di dialogo macro o il tasto di scelta rapida o in VBE o assegnando la macro a un oggetto. La finestra di dialogo macro raggiungibile dalla scheda *Visualizza* della barra multifunzione, cliccare su *macro* e poi su *Visualizza macro* che aprirà la finestra di dialogo macro, in cui si deve selezionare il nome della macro e cliccare sul pulsante *Esegui* per eseguire la Sub/macro. È inoltre possibile aprire la finestra di dialogo macro facendo clic su *Macro* nel gruppo Codice della scheda Sviluppo sulla barra multifunzione e scegliere *Visual Basic Editor*

- In Visual Basic Editor, fare clic all'interno del sub e premere F5 (o fare clic su Esegui nel barra dei menu e quindi fare clic su Esegui Sub/UserForm).
- In Visual Basic Editor, fare clic su Strumenti nella barra dei menu, quindi fare clic su macro che apre la finestra di dialogo Macro. Nella finestra di dialogo Macro, selezionare il nome della macro e fare clic su Esegui per eseguire la macro.

Un buon modo per eseguire una macro potrebbe essere quella di fare clic su un pulsante accompagnato da un testo esplicativo che appare sul foglio di lavoro, per questo è necessario assegnare la macro a un oggetto, forma, grafico o un controllo. È possibile assegnare una macro a qualsiasi controllo modulo agendo dalla scheda Sviluppo sulla barra multifunzione, cliccando su *Inserisci* nel gruppo Controlli, selezionare e fare clic sul pulsante nei controlli modulo e quindi fare clic sul foglio di lavoro in cui si desidera posizionare l'nell'angolo superiore sinistro del pulsante (è possibile spostare e ridimensionare in seguito).

Successivamente cliccando col destro del mouse sul pulsante e dal menu a discesa che compare selezionare *Assegna macro* e si apre la finestra di dialogo Assegna macro dal quale è possibile selezionare e assegnare una macro all'oggetto disegnato. E' possibile assegnare macro a qualsiasi oggetto, forma, immagine, grafico etc. inserito nel foglio di lavoro.

Proseguimento linea del codice VBA

Molte volte durante la scrittura di codice VBA, una sola riga di codice potrebbe diventare molto lunga e nella finestra del codice sarà necessario scorrere verso il lato destro per leggerlo. Per dividere una singola riga in più righe in VBA, è possibile utilizzare la linea come carattere di continuazione che è la combinazione di uno spazio seguito da un underscore (_), usando questo carattere come una interruzione di linea, vi permetterà di passare alla riga successiva e così via, e questo sarà il trattamento di tutte le linee continue come una singola riga nel codice VBA.

Si noti che si può avere un massimo di 25 linee fisicamente, unite con il carattere di continuazione di riga (cioè un massimo di 24 caratteri di continuazione della riga), e di essere trattate come una singola riga logica di codice. Una linea fisica può avere al massimo 1023 caratteri mentre una linea logica può avere un massimo di 10.230 caratteri, in modo che un massimo di 25 linee fisiche possono essere unite pari ad un massimo di 10.230 caratteri.

Controllo Automatico Sintassi

Durante la scrittura di codice VBA, il codice che ha un errore di sintassi diventa rosso non appena il cursore si sposta sulla linea se il controllo automatico di sintassi è selezionato (impostazione di default), un controllo della sintassi viene eseguito ogni volta che si sposta il cursore su una nuova linea, e in caso di errore di sintassi una finestra pop-up avverte con un messaggio "*errore di compilazione*". Se il controllo automatico di sintassi è deselezionato, non sarà più possibile ottenere questi box di avviso, anche se l'errore farà diventare di colore rosso la riga del codice. È possibile deselezionare il controllo automatico di sintassi andando in VBE, seguendo il percorso ***Strumenti – Opzioni*** e nella finestra visualizzata, selezionare la scheda Editor, e quindi deselezionare la casella di controllo.

È inoltre possibile modificare il colore rosso di default dell'errore di sintassi andando in VBE, cliccate su ***Strumenti – Opzioni*** e nella finestra visualizzata, selezionare la scheda Formato, selezionare sintassi del testo di errore nella casella di riepilogo codice colori, e poi scegliere il nuovo colore di primo piano.

Commento del codice in VBA

All'interno di una procedura, durante la scrittura di codice è possibile contemporaneamente fornire commenti per spiegare lo scopo e ciò che il codice sta facendo. Per differenziare i commenti con il codice, dovete inserire un singolo carattere di apostrofo (') o con la dicitura Rem seguito da uno spazio. I commenti verranno ignorati da VBA in modo che un errore di sintassi nel commento non viene restituito. Si consiglia di utilizzare i commenti nel codice che saranno di grande aiuto a un altro utente nella comprensione, o se si deve effettuare una modifica in un secondo momento. Una volta che si aggiunge un carattere di commento (') all'inizio di una riga è possibile utilizzare la sequenza di continuazione (_) per continuare il commento alla riga successiva e quando si preme il tasto Invio dopo aver digitato un commento, il suo colore diventa verde, come da impostazione predefinita di VBA.

Questa impostazione predefinita del colore del commento può essere modificato andando in VBE, e seguendo il percorso ***Strumenti – Opzioni*** e nella finestra visualizzata selezionare la scheda Formato, selezionare il commento del testo nella casella di riepilogo Colori e quindi scegliere il nuovo colore di primo piano. Si noti che il testo di commento non deve necessariamente iniziare all'inizio di una riga, può essere inserita sul lato destro in una riga di codice, lasciando pochi spazi dopo il codice e quindi digitando un singolo apostrofo seguito da commento.

Indentare il codice VBA

La procedura di VBA può essere composto da più righe di codice con una serie di dichiarazioni e quando il codice diventa più lungo e complesso, la formattazione del codice con un rientro contribuirà a rendere più facile da leggere, e renderà il debug più semplice. Gli sviluppatori utilizzano in genere un rientro per il codice all'interno delle linee di inizio e fine dei Loop in cui una serie di righe di codice o in un ciclo For Next si usa rientrare per distinguerli come appartenenti ad un particolare procedimento. L'indentazione è molto utile nel codice nidificato in modo che ogni blocco di codice è visivamente separato, e le linee di inizio e fine di ogni blocco sono allineate.

L'Indentazione di solito è fatta premendo il tasto Tab una o più volte, prima di digitare il codice. Si noti che per impostazione predefinita in VBA, premendo il tasto Tab si sposta il cursore di quattro spazi o caratteri a destra. Questa impostazione del valore della scheda può essere modificata andando a VBE, e seguendo il percorso ***Strumenti – Opzioni*** e nella finestra visualizzata selezionare la scheda Editor e inserire il nuovo valore nella casella "*Larghezza linguetta*". Nella scheda Editor della finestra di dialogo Opzioni, è possibile anche selezionare "*Rientro tabulazione*", che ripeterà il rientro della riga corrente premendo Invio.

Utilizzo di variabili in VBA

Una variabile è una posizione di memoria utilizzata per memorizzare valori temporanei o informazioni per l'uso in esecuzione del codice e in un programma VBA, il contenuto delle variabili viene utilizzato o modificato successivamente durante l'esecuzione del codice. Dichiarando una variabile da utilizzare nel codice, si indica al compilatore di Visual Basic il tipo di dati contenuto nella variabile (Integer, Testo, Boolean, etc.) e altre informazioni come la sua visibilità (Public o Private).

Le variabili devono essere esplicitamente dichiarate usando le parole chiave Dim, Private, Public, ReDim o dichiarazioni statiche e quando si dichiarano utilizzando un'istruzione Dim (Dim è l'abbreviazione di dimensione): per dichiarare una variabile per contenere un valore intero, usare "*Dim riga1 As Integer*"; per dichiarare una variabile per contenere valori di testo, usare "*Dim riga2 As String*"; e così via.

Parole chiave di VBA

Le parole chiave sono parole riservate che VBA usa come parte del suo linguaggio di programmazione e sono parole o comandi che sono riconosciuti da VBA e possono essere utilizzati nel codice solo come parte del linguaggio VBA (come in una dichiarazione, il nome della funzione, o l'operatore) e non altrimenti (come i nomi delle sub-routine o variabile). Esempi di parole chiave sono: Sub, End, Dim, If, Next, And, Or, Loop, Do, Len, Close, data, Else, Select, e così via. Per ottenere aiuto su una determinata parola chiave, inserire il cursore del mouse all'interno della parola (nel codice VBA in VBE) e premere F1.

Operatori aritmetici

In VBA, è possibile eseguire calcoli con valori numerici utilizzando gli operatori aritmetici, si utilizza il segno "**+**", per aggiungere i valori, il segno "*****" per la moltiplicazione e il segno "**-**" per la sottrazione. Per la divisione si utilizza il segno "**/**", per dividere due valori mentre per divisioni tra interi, utilizzare il segno contrapposto "\", dove sia il dividendo che il divisore devono essere numeri interi e il risultato sarà un numero intero ignorando la parte decimale che significa che la divisione di interi restituisce il quoziente e ignora qualsiasi residuo.

Per esempio, quando si divide 7 per 3, 7 si chiama dividendo e 3 il divisore, il quoziente è 2, e il resto è 1, la divisione intera restituirà il numero 2 e ignora il resto. Per esponenziali si utilizza l'operatore "^", per aumentare la potenza di un numero ad un altro numero corrispondente alla moltiplicazione ripetuta, è inoltre possibile utilizzare le parentesi con gli operatori aritmetici, ad esempio, se si desidera aggiungere prima 5 e 3 e poi moltiplicare il risultato per 7, si digita (5 + 3) * 7, oppure con 5 + 3 * 7.

Operatori di concatenazione

In VBA, l'operatore di concatenazione è rappresentato dalla e commerciale **(&)** e viene utilizzata per concatenare più stringhe in una singola stringa. Se si prendono 2 stringhe "Ex" e "cel", si utilizza l'operatore di concatenazione (&), per ottenere una singola stringa "Excel". ("Ex" e "cel" = "Excel"). L'operatore di concatenazione è spesso usato in VBA, per unire o collegare più stringhe di testo in una sola stringa di testo.

Utilizzare la funzione MsgBox nel codice

La funzione ***MsgBox*** viene spesso utilizzata nel codice VBA per visualizzare un messaggio in una finestra di dialogo, in cui è richiesta la risposta dall'utente facendo clic su un pulsante apposito (Ok, Annulla, Sì, No, Riprova, Ignora o Annulla). Una finestra di messaggio viene anche comunemente usata come uno strumento di debug, per convalidare o controllare il codice da eventuali errori ed è un mezzo per interagire con l'utente, per visualizzare un valore restituito eseguendo un'istruzione o un codice, o se si desidera che un'azione venga confermata da parte dell'utente prima di essere eseguita, come cancellare o salvare qualcosa, o se si vuole consentire all'utente di sapere che la macro ha terminato l'esecuzione, e così via.

Il codice più semplice per la visualizzazione di una finestra di messaggio è con l'istruzione MsgBox "Ciao" che quando viene eseguito il codice, una finestra di dialogo apparirà e visualizzerà il messaggio "Ciao" con il pulsante "OK", cliccando sul quale il messaggio sarà chiuso e l'esecuzione di codice continuerà. Il messaggio che si desidera visualizzare nella finestra di messaggio deve essere digitato tra virgolette vale a dire. "Ciao"

Esempio: Scrivere il testo nel Range, cambiando il font e il colore di fondo della cella

```
Sub prova1 ()
'Inserire il testo "ciao" in A1: C5 di "Foglio1"
ThisWorkbook.Worksheets("Foglio1").Range("A1:C5").Value = "Ciao"
`impostare il tipo di carattere
ThisWorkbook.Worksheets("Foglio1").Range("A1:C5").Font.Name = "Times New Roman"
ThisWorkbook.Worksheets("Foglio1").Range("A1:C5").Font.Size = 12
ThisWorkbook.Worksheets("Foglio1").Range("A1:C5").Font.Italic = True
ThisWorkbook.Worksheets("Foglio1").Range("A1:A5").Font.Bold = True
` impostare il colore di sfondo della cella
ThisWorkbook.Worksheets("Foglio1").Range("A1:A5").Interior.ColorIndex = 6
` impostare il colore del carattere
ThisWorkbook.Worksheets("Foglio1").Range("B1:B5").Font.ColorIndex = 5
'impostare il colore del font
ThisWorkbook.Worksheets("Foglio1").Range("C1:C5").Font.ColorIndex = 3
End Sub
```

Esempio: Scrivere lo stesso codice come sopra utilizzando altri comandi

```
Sub prova2 ()
With ThisWorkbook.Worksheets("Foglio1")
With .Range("A1:C5")
.Value = "Ciao"
With .Font
.Name = "Times New Roman"
.Size = 12
.Italic = True
End With
End With
With .Range("A1:A5")
.Font.Bold = True
.Interior.ColorIndex = 6
End With
.Range("B1:B5").Font.ColorIndex = 5
.Range("C1:C5").Font.ColorIndex = 3
End With
End Sub
```

Esempio: Fare calcoli in VBA con gli operatori aritmetici

```
Sub prova5 ()
'operiamo su ThisWorkbook
With ThisWorkbook
`aggiungiamo un nuovo foglio dopo l'ultimo foglio di lavoro
Worksheets.Add After:=Worksheets(Worksheets.Count)
'Il nuovo foglio di lavoro diventa il foglio attivo
With .ActiveSheet
. Range ("A1"). Value = "Punteggio"
. Range ("B1"). Value = "Punteggio inglese"
. Range ("C1"). Value = "Medio"
'impostare e formattare il formato numerico del range
.Range("A2:C2").NumberFormat = "#,##0.00"
.Range("A2").Value = 57
.Range("B2").Value = 84
'calcolare la media
.Range("C2").Value = (.Range("A2").Value + .Range("B2").Value) / 2
'utilizzare il metodo Adatta per le colonne A: C
.Columns("A:C").AutoFit
'inserire il nuovo nome del foglio di lavoro in A4
.Range("A4").Value = "Il Nome del foglio di lavoro è: " & .Name
'si porta a video la media
MsgBox "Il punteggio Medio è: " & .Range("C2").Value
End With
End With
End Sub
```

Esempio: Assegnazione di un oggetto ad una variabile

```
Sub prova7 ()
Dim var1 As Range, var2 As Range, var3 As Range, var4 As Range
Set var1 = ThisWorkbook.Worksheets("Foglio1").Range("A1:C5")
Set var2 = ThisWorkbook.Worksheets("Foglio1").Range("A1:A5")
Set var3 = ThisWorkbook.Worksheets("Foglio1").Range("B1:B5")
Set var4 = ThisWorkbook.Worksheets("Foglio1").Range("C1:C5")
With var1
.Value = "Ciao"
With .Font
.Name = "Arial"
.Size = 12
.Italic = True
End With
End With
With var2
.Font.Bold = True
.Interior.ColorIndex = 6
End With
var3.Font.ColorIndex = 5
var4.Font.ColorIndex = 3
End Sub
```

Esempio: Scrivere lo stesso codice come sopra utilizzando le variabili

```
Sub prova6 ()
Dim n As Double
With ThisWorkbook
Worksheets.Add After:=Worksheets(Worksheets.Count)
With .ActiveSheet
. Range ("A1"). Value = "Punteggio"
. Range ("B1"). Value = "Punteggio inglese"
. Range ("C1"). Value = "Medio"
.Range("A2:C2").NumberFormat = "#,##0.00"
.Range("A2").Value = 57
.Range("B2").Value = 84
'calcolare la media
n = (.Range("A2").Value + .Range("B2").Value) / 2
.Range("C2").Value = n
.Columns("A:C").AutoFit
.Range("A4").Value = "Il Nome del foglio di lavoro è: " & .Name
MsgBox "Il punteggio Medio è: " & .Range("C2").Value
End With
End With
End Sub
```

```
Sub prova4 ()
ThisWorkbook.Activate                'Attiva ThisWorkbook
Worksheets ("Foglio1"). Activate     'attiva "Foglio1"
With ActiveSheet                 'Operiamo nel foglio attivo
With .Range("A1")                'Operiamo sulla cella A1
.Value = "Prova VBA"             'Inseriamo del testo in A1
.Interior.Color = vbYellow       'impostiamo il colore di fondo della cella A1 a giallo
.Font.Bold = True                'Impostare il font a grassetto in A1
. HorizontalAlignment = xlCenter   `impostare l'allineamento orizzontale in A1
End With
'in A2 si inserisce il testo e il nome della cartella attiva
.Range("A2").Value = " Il nome della cartella attiva è: " & ActiveWorkbook.Name
'inserire il nome foglio attivo nella cella A3
.Range("A3").Value = "Il nome del foglio attivo è: " & ActiveSheet.Name
'attivare la cella A4
.Range("A4").Activate
'inserire l'indirizzo di cella in A4
.Range("A4").Value = "L'indirizzo della cella attiva è: " & ActiveCell.Address
'operiamo sulla colonna A del foglio attivo
With Columns("A")
'operiamo sul Font nella colonna A
With .Font
. Name = "Arial"       'impostiamo il nome del carattere
. Size = 10            'impostiamo la dimensione del carattere
. Color = vbBlue       'impostiamo il colore del testo
End With
.AutoFit               'Adattiamo il Range alla larghezza della colonna A
End With
End With
End Sub
```

Esempio: Utilizzare una finestra di messaggio per restituire il numero di cartelle di lavoro aperte e i loro nomi

```
Sub prova3 ()
'Conta il numero di cartelle di lavoro aperte
MsgBox Workbooks.Count
'conta il numero di fogli di lavoro in ThisWorkbook
MsgBox ThisWorkbook.Worksheets.Count
'restituisce il nome di ThisWorkbook
MsgBox ThisWorkbook.Name
'restituisce la cartella di lavoro attiva
MsgBox ActiveWorkbook.Name
'restituisce il foglio attivo
MsgBox ActiveSheet.Name
End Sub
```