Azioni ripetitive: Il Ciclo For e il ciclo For Each

Ora che sappiamo come scegliere diverse azioni basandosi su delle condizioni predeterminate, siamo pronti a vedere come ripetere delle azioni un numero predeterminato di volte se si verifica, o non si verifica una particolare condizione nella nostra procedura. Uno degli svantaggi delle macro è la loro incapacità di ripetere le azioni a meno che le azioni desiderate non vengano registrate ripetutamente, VBA fornisce diverse strutture potenti e versatili per permetterci di ripetere facilmente le azioni e per controllare il modo in cui VBA effettua queste ripetizioni.

Le strutture del programma che ripetono l'esecuzione di una o più istruzioni sono chiamate **Cicli**, alcune strutture per i cicli sono costruite in modo da venire eseguite un numero impostato di volte, e vengono chiamate *cicli ad interazione fissa*, mentre altri tipi di strutture per i cicli vengono impostate un numero variabile di volte sulla base di alcune condizioni impostate, proprio perchè il numero di ripetizioni di queste strutture non è definito questi cicli vengono chiamati *cicli ad interazione indefinita*.

Sia nelle strutture ad interazione fissa che nelle strutture ad interazione indefinita, ci sono alcune espressioni che determinano quante volte il ciclo viene ripetuto, questa espressione viene chiamata *determinante del ciclo*, questa espressione in un ciclo ad interazione fissa è quasi sempre una espressione numerica, mentre per i cicli a interazione indefinita la determinante del ciclo è solitamente un'espressione logica che descrive la condizione sotto la quale il ciclo può continuare o interrompere la sua esecuzione, praticamente vengono usate delle espressioni logiche per la determinante dei cicli allo stesso modo in cui sono state usate per prendere delle decisioni in VBA.

Il ciclo For Next

La più semplice struttura per i cicli è quella ad interazione fissa, VBA ne fornisce due diverse tipologie che vengono espresse così For ...Next e For ...Each ... Next entrambi i cicli vengono chiamati *cicli For* perchè vengono eseguiti per uno specifico numero di volte. Il ciclo For ... Next ha la seguente sintassi

```
For contatore = inizio To fine
istruzioni
Next contatore
```

contatore viene rappresentato da una qualsiasi variabile numerica, di tipo *Integer* o *Long*, *inizio* è anch'esso rappresentato da una variabile numerica e specifica il valore iniziale per la variabile *contatore*, anche *fine* è una espressione numerica che rappresenta il valore finale per la variabile *contatore*. Per default VBA incrementa la variabile [I]contatore di **1** ogni volta che esegue le istruzioni di un ciclo. La parola chiave **Next** indica a VBA che è stata raggiunta la fine del ciclo, inoltre la variabile *contatore*, dopo la parola chiave *Next*, deve essere la stessa variabile *contatore* che abbiamo messo appena dopo l'enunciato For all'inizio della struttura del ciclo.

Quando VBA esegue un ciclo For, prima assegna il valore rappresentato da inizio alla variabile *contatore*, quindi esegue tutte le istruzioni rappresentate da *istruzioni* fino a quando non raggiunge la parola chiave *Next* che indica che è stata raggiunta la fine del ciclo. VBA poi incrementa la variabile *contatore* di 1 e ritorna all'inizio del ciclo e confronta il valore corrente della variabile *contatore* col valore rappresentato da *fine*. Se *contatore* è minore o uguale a *fine* esegue nuovamente il ciclo. Se invece *contatore* è maggiore di *fine* VBA esce dal ciclo e continua la sua esecuzione con la prima istruzione dopo la parola chiave *Next*.

Possiamo però anche specificare un valore diverso per l'incremento della variabile contatore includendo la parola chiave **Step** [opzionale], in questo caso dobbiamo specificare l'incremento della variabile *contatore*, e la sintassi diventa così:

For contatore = inizio To fine Step passo
istruzioni
Next contatore

In questo caso l'espressione *passo* viene rappresentato da una espressione numerica e indica la quantità per incrementare la variabile *contatore*, vediamo qualche esempio

```
Sub for_semplice()
 For I = 1 To 10
 MsgBox (I)
 Next I
End Sub
```

L'esecuzione di questo codice ci porta a video la finestra di messaggio (MsgBox) per 10 volte con il valore della variabile I (*contatore*), se invece vogliamo usare la parola chiave *Step* e far apparire solo le espressioni dispari possiamo modificare il listato in questo modo

```
Sub for_step()
 For I = 1 To 10 Step 2
 MsgBox (I)
 Next I
End Sub
```

E ci verranno riportati i soli valori dispari della variabile nella finestra di dialogo di MsgBox. Con la parola chiave *Step*, abbiamo modificato l'incremento della variabile da **1** (di default) a **2**, in pratica alla prima esecuzione del ciclo **I** *vale 1* ma quando incontra la parola chiave *Step* viene incrementata di **2**, se fate "girare" la macro for_step noterete che al primo ciclo viene stampato il valore 1, questo perchè **I** quando incontra la parola chiave *Step* vale ancora 1, per cambiare valore deve arrivare alla parola chiave *Next* (che abbiamo detto essere quella che avvisa di essere arrivati alla fine del ciclo e che incrementa la variabile).

In ultima analisi l'enunciato For viene interpretato così : *For I* [Per I] = *1 To 10* [che và da 1 a 10] *Step 2* [incrementa di 2], *Esegui le istruzioni , Next I*[incrementa il valore di I], Il ciclo *For ..Next* ha la flessibilità di poter incrementare e decrementare la variabile, possiamo modificare il listato da *For I = 1 To 10 Step 2* a *For I = 100 To 1 Step -2* pertanto le possibilità di impiego sono abbastanza vaste, possiamo eseguire somme, incrementare e decrementare il valore delle variabili, utilizzare la ciclicità per ogni bisogno che richieda il nostro programma fermo restando i principi di impiego esposti all'inizio. Esiste anche un'altra forma di ciclo ed è il ciclo *For Each ..Next*

Il ciclo For Each

A differenza del ciclo *For … Next* il ciclo **For Each ... Next** non usa un contatore, ma viene eseguito tante volte quanti sono gli elementi di un gruppo specifico, come una collezione di oggetti o un vettore. In breve il ciclo For Each … Next viene eseguito una volta per ogni elemento di un gruppo. Questo tipo di ciclo ha la seguente sintassi

For Each elemento In gruppo
Istruzioni
Next [elemento]

Dove, *elemento* è una variabile usata per interare il ciclo per tutti gli elementi del gruppo, *gruppo* è una collezione di oggetti o un vettore (matrice), se *gruppo* è una collezione di oggetti, *elemento* deve essere una variabile di tipo *Variant* o *Object* o uno specifico tipo di oggetto come *Range*, *Worksheet*, *Cells*e così via. *Istruzioni* rappresenta nessuna, una o più istruzioni VBA che formano il corpo del ciclo. Questo tipo di ciclo ha meno opzioni del ciclo For Next, l'incremento del contatore non è rilevante in questo ciclo, perchè viene sempre eseguito tante volte quanti sono gli elementi presenti nel gruppo specificato. In pratica non dovendo indicare il numero iniziale e finale per impostare quante volte il ciclo deve ripetere le nostre istruzioni, la nostra ricerca sarà rappresentata da quante volte il dato da cercare sarà presente nell'area di ricerca e l'inizio sarà sempre dal primo record contenuto nell'area di ricerca.

Per meglio comprendere vediamo un esempio di questo tipo : Supponiamo di avere un elenco di nomi nell'intervallo di celle (Range) A1:A5 e si deve confrontare il valore della cella B1 e verificare se è presente nel Range, possiamo scrivere questo codice :

```
Sub prova_each()
Dim x As Boolean
y = Range("B1").Value
  For Each cl In Range("A1:A5")
   If cl = y Then
   x = True
  End If
 Next
If x = True Then
  MsgBox ("Valore Presente")
Else
  MsgBox ("Valore Assente")
End If
End Sub
```

In questo listato *elemento* è rappresentato dalla variabile **y**, cioè dal valore contenuto nella cella B1 e *gruppo* è rappresentato dal Range A1:A5, mentre il corpo del ciclo è rappresentato dalle istruzioni MsgBox se il valore è presente oppure no. In questo tipo di ciclo il numero iniziale da cui partire è definito dalla prima cella del Range cioè A1 e il numero finale dall'ultima cella dello stesso Range, cioè A5. Adesso il ciclo sa dove iniziare, dove finire e cosa cercare (cerca il valore della cella B1) e scorrendo tutte le celle del Range esegue le istruzioni MsgBox se il valore di B1 è presente oppure no nel percorso di confronto Possiamo quindi semplificare affermando che gruppo può essere un insieme di oggetti come le celle, le righe e le colonne di un foglio di calcolo, i fogli di lavoro di un file Excel oppure gli oggetti che possiamo inserire in un foglio come pulsanti, caselle di testo pulsanti di comando Userform etc..

E' indubbio che un ciclo For Each rappresenta un modo pratico per sfogliare gli insiemi di una collezione, va ricordato che l'istruzione For Each funziona come Set, ovvero assegna un riferimento dell'oggetto a una variabile, ma invece di assegnarne uno solo, sceglie tutti gli elementi di una raccolta. Quindi, per ogni oggetto della raccolta, Visual Basic esegue tutte le istruzioni fino all'istruzione Next, anche se tecnicamente, non è necessario inserire il nome della variabile dopo Next, se lo si utilizza, Visual Basic richiede che corrisponda al nome della variabile dopo For Each. Si consiglia di utilizzare sempre la variabile del ciclo dopo Next in modo che Visual Basic possa contribuire a evitare bug nella macro. Le istruzioni che hanno inizio con For Each e terminano con Next sono definite blocchi For Each o cicli For Each.

Cicli Nidificati

È possibile nidificare cicli mettendo un ciclo all'interno di un altro ciclo, fino a un numero illimitato di volte. La variabile contatore per ogni ciclo deve essere univoca ed è possibile nidificare un tipo di ciclo all'interno di un altro tipo diverso. In un ciclo for, è necessario che il ciclo interno sia completato prima che si esegua l'istruzione successiva del ciclo esterno. È inoltre possibile nidificare un tipo di struttura di controllo all'interno di un altro tipo vale a dire che è possibile nidificare un'istruzione IF all'interno di un blocco With che può a sua volta essere annidato all'interno di un For ... Each. Tuttavia, le strutture di controllo non possono essere sovrapposte, ogni blocco nidificato si deve chiudere e terminare entro il livello nidificato esterno.

Esempio di nidificazione di un ciclo IF all'interno di un blocco With che è annidato all'interno di un For. Il codice del ciclo Loop scorre ogni cella nell'intervallo A1: A10, e se il valore della cella è superiore a 5, il colore di sfondo viene impostato come giallo, mentre per valori inferiori a 5 viene usato il colore rosso

```
Sub ciclo1()
Dim iCell As Range
For Each iCell In ActiveSheet.Range("A1:A10")
With iCell
If iCell > 5 Then
.Interior.Color = RGB(255, 255, 0)
Else
.Interior.Color = RGB(255, 0, 0)
End If
End With
Next iCell
End Sub
```

Uscita anticipata dal ciclo

È possibile uscire da un ciclo For (For ... Next e For Each ... Next) in anticipo, senza completare il ciclo, utilizzando la dichiarazione Exit For che interromperà immediatamente l'esecuzione del ciclo esistente e passerà ad eseguire la sezione di codice immediatamente successiva all'istruzione Next, e nel caso di livello nidificato interno si ferma ed eseguirà il successivo livello nidificato esterno. Si può avere qualsiasi numero di istruzioni Exit For in un ciclo ed è particolarmente utile nel caso in cui si desidera terminare il ciclo al raggiungimento di un certo valore o di soddisfare una condizione specifica, o nel caso in cui si desidera interrompere un Loop infinito a un certo punto.

Esempio: Se la cella A1 è vuota, la variabile nTotal si somma al valore 25, mentre se la cella A1 contiene il valore 5, il ciclo termina ed esce quando il contatore raggiunge il valore 5

```
Sub ciclo2 ()
Dim n As Integer, nTotal As Integer
nTotal = 0
For n = 1 To 10
nTotal = n + nTotal
If n = ActiveSheet.Range("A1") Then
Exit For
End If
Next n
MsgBox nTotal
End Sub
```