Azioni ripetitive: Il Ciclo Do Loop

Abbiamo parlato di due tipi di cicli, quelli ad *interazione fissa* e quelli ad *interazione indefinita*, il ciclo **Do ..Loop** appartiene ai cicli ad interazione indefinita, VBA ci fornisce questa istruzione estremamente potente per costruire strutture cicliche indefinite nelle nostre funzioni o procedure. Essenzialmente è costituito da una singola istruzione: Do. Questa istruzione ha molte opzioni ed è talmente flessibile che ci fornisce quattro diverse possibilità per costruire dei cicli raggruppati in due diverse categorie di base, che sono i cicli controllati da un contatore e i cicli controllati da eventi, quale è la differenza?

In un ciclo controllato da un contatore, le istruzioni del corpo del ciclo vengono eseguite finchè il valore è inferiore o superiore al limite specificato, in sostanza non varia molto dal ciclo For ..Next, eccetto che il programmatore è direttamente responsabile per l'inizializzazione della variabile contatore e per l'incrementazione o decrementazione del contatore. Potremmo usare il ciclo Do se il passo del contatore non è regolare, o se non c'è modo di determinare il limite finale se non dopo che il ciclo ha iniziato la sua esecuzione. Per esempio se vogliamo spostarci attraverso 15 righe di un foglio, alcune volte avanzando di una sola riga e altre volte avanzando di due righe, poichè il numero di righe da avanzare (cioè il passo del contatore) cambia, non possiamo usare il ciclo *For ..Next* ma dobbiamo usare il ciclo Do

Per i cicli controllati da eventi, le istruzioni vengono eseguite quando la determinante del ciclo diventa vera o falsa sulla base di alcuni eventi che si verificano all'interno del corpo del ciclo. Per esempio potremmo scrivere un ciclo che viene eseguito indefinitamente fino a quando l'utente non inserisce un particolare valore in una finestra di dialogo input, e l'inserimento di questo particolare valore è l'evento che termina il ciclo, oppure possiamo eseguire delle operazioni sulle celle di un foglio fino a quando non si raggiunge la cella vuota di una colonna, anche in questo caso il raggiungimento della cella vuota è l'evento che termina il ciclo. Per ora abbiamo chiarito le due categorie di cicli, abbiamo capito che esistono cicli controllati da un contatore e cicli controllati da eventi, vediamo ora la sintassi

```
Do
istruzioni
Loop Until condizione
```

e qualche esempio.

```
Sub ciclo1()
Dim a As Integer
Do
 a = a + 1
 MsgBox (a)
Loop Until a = 10
End Sub
```

Questo è un ciclo controllato da un contatore, noterete che c'è poca differenza dal ciclo For ..Next , infatti otteniamo lo stesso effetto, cioè portiamo a video un messaggio col valore della variabile finchè non arriva a 10, ma abbiamo visto poco sopra che abbiamo quattro diverse possibilità di costruire cicli divisi in due categorie, ora le categorie le abbiamo viste *(cicli controllati da un contatore e i cicli controllati da eventi)* vediamo ora i quattro modi di costruire un ciclo, il listato sopra esposto è un metodo, vediamo ora gli altri e dopo li commentiamo assieme

```
Sub ciclo2()
Dim a As Integer
Do Until a = 10
 a = a + 1
 MsgBox (a)
Loop
End Sub

Sub ciclo3()
Dim a As Integer
Do
a = a + 1
MsgBox (a)
Loop While a <> 10
End Sub

Sub ciclo4()
Dim a As Integer
Do While a <> 10
a = a + 1
MsgBox (a)
Loop
End Sub
```

Come potete vedere la differenza sta nell'enunciato di dichiarazione del ciclo, il risultato non cambia ma le parole chiave e la loro collocazione si, vediamo come viene interpretato l'enunciato : *Do Until a = 10* [Ripeti finchè a = 10], oppure *Do While a <> 10* [Ripeti finchè a è diverso da 10], in questi 2 casi la differenza fondamentale è che viene verificata la condizione determinante *prima* che venga eseguito il ciclo, infatti se a fosse = a 20 il ciclo non verrebbe eseguito, gli altri 2 metodi Do .....Loop While a <> 10 e Do ..... Loop Until a = 10 vengono interpretati come spiegato sopra, ma la determinate del ciclo viene verificata alla fine del ciclo.

In pratica la forma *Do ... Loop While* e *Do ... Loop Until* prima vengono eseguite le istruzioni presenti nel ciclo e poi quando raggiunge la parola chiave *Loop* viene verificata la condizione (vedi sintassi) se condizione è *False* (nel caso si usi Until VBA ritorna all'inizio del ciclo ed esegue nuovamente le istruzioni del ciclo, se invece usiamo la forma While la condizione da verificare deve essere *True*, mentre nelle altre 2 forme espresse Do Until ...Loop e Do While ...Loop la condizione invece viene verificata subito.

Vediamo un esempio che semplifica e chiarifica quanto esposto.
Supponiamo di far comparire a video un Inputbox per chiedere informazioni all'utente, e poi verificare se i dati immessi siano validi, in questo caso inseriamo nel nostro ciclo dobbiamo eseguire le istruzioni (far comparire il messaggio di Input) e all'inserimento dei dati da parte dell'utente verifichiamo la condizione, il listato si presenta così

```
Sub esempio1()
Dim pass As String
Do
pass = InputBox(prompt:="Inserisci password di accesso per Proseguire: ",
Title:="Controllo accessi")
Loop Until pass = "Alex"
MsgBox "Password esatta: Accesso consentito", vbInformation + vbYes, "Verifica Password"
End Sub
```

Se proviamo questo codice vedremmo che il ciclo continua a ripetere le istruzioni finchè non viene digitata la password esatta, però se vogliamo uscire perchè non ricordiamo la password?, anche premendo sul tasto "Annulla" dell'Inputbox le istruzioni continuano ad essere ripetute lo stesso, allora è sempre meglio porre una condizione per evitare di proseguire, ma al tempo stesso controllare che l'esecuzione del ciclo avvenga correttamente, in pratica mettiamo l'utente in grado di uscire da un ciclo senza però che eviti di inserire la password richiesta. Per ovviare a questo modificchiamo il listato in questo modo.

```
Sub esempio2()
Dim pass As String
Do
pass = InputBox(prompt:="Inserisci password di accesso per Proseguire: ",
Title:="Controllo accessi")
If pass <> "Alex" Then Exit Sub
Loop Until pass = "Alex"
MsgBox "Password esatta : Accesso consentito", vbInformation + vbYes, "Verifica
Password"
End Sub
```

Inserendo la riga con il ciclo **IF** abbiamo posto una condizione, cioè *"Se pass è diverso da "Alex" esci dalla sub"*, attenzione, che esci dalla sub non vuol dire proseguire, ma semplicemente abbandoniamo questa routine che in ogni caso per poter proseguire dobbiamo inserire la password giusta, questa semplice metodica viene chiamata *Uscita forzata dal ciclo".*
Anche con l'altro metodo cioè *Do Until ...Loop* e *Do While ...Loop* il risultato non cambia, pertanto non c'è una regola ben precisa su quale forma sia meglio usare, possiamo dire che a disposizione VBA ci mette queste forme, sta a noi usare quella che ci pare più logica o intuitiva per esprimere quello che vogliamo esegua il nostro codice. Se vi ricordate nella lezione precedente avevamo parlato della nidificazione del ciclo IF, anche nei cicli For ... Next e Do ...Loop è possibile eseguirla, visto che l'argomento lo abbiamo già toccato in questo contesto facciamo subito un esempio e dopo lo commenteremo assieme

```
Sub scrivi_col_for()
For riga = 12 To 35
 For colonna = 7 To 18
 Cells(riga, colonna).Value = riga
 Next colonna
Next riga
End Sub
```

Cosa abbiamo fatto? se provate ad eseguire la macro vedrete che nell'area di lavoro abbiamo riempito tutte le celle con il numero della riga corrispondente. Vediamo il codice *For riga = 12 To 35* il contatore del ciclo è rappresentato dalla variabile riga e gli diciamo *[Per riga che và da 12 a 35]* e subito dopo invece di far eseguire le istruzioni gli mettiamo un'altro enunciato *For colonna = 7 To 18* in questo caso il contatore del ciclo è rappresentato dalla variabile colonna e lo interpretiamo così *[Per colonna che và da 7 a 18]* , a questo punto abbiamo posto due condizioni una sotto l'altra e subito dopo abbiamo posto le istruzioni *Cells(riga, colonna).Value = riga* in questo enunciato la parola chiave *Cells* (che vedremmo nella prossima lezione) sta ad indicare una determinata cella del nostro foglio localizzata dai valori delle variabili riga e colonna, l'altra parola chiave *Value* indica il valore da inserire nelle coordinate rappresentate, tale valore lo abbiamo identificato con riga.

Seguendo l'esecuzione del ciclo, prima viene eseguito il ciclo interno, e cominciamo dalla cella che si trova all'intersezione tra la colonna n° 7 e la riga n° 12che sul nostro foglio è rappresentata da **G12**, al cui interno scriviamo il valore di riga (per cui 12), poi incontriamo la parola chiave *Next colonna*, a questo punto però non abbiamo ancora raggiunto la determinante del ciclo (rappresentato dal valore [18]) e VBA incrementa il contatore di 1 e ripete le istruzioni, così facendo andiamo a scorrere tutte le colonne e scriviamo al loro interno il valore della variabile riga.

Una volta raggiunta la determinante del ciclo usciamo dal ciclo interno ma troviamo la parola chiave *Next riga*, per cui il contatore del ciclo esterno viene incrementato, passiamo alla riga successiva (la 13) e ripetiamo il ciclo interno come abbiamo descritto sopra. Il nostro listato ha fine quando viene raggiunta la determinate del ciclo esterno e a questo punto troveremo la nostra area di lavoro riempita con il valore della variabile riga estesa sulle colonne di ciascuna riga. Ora che abbiamo riempito l'area con il valore della riga tramite un ciclo For Next nidificato potremmo vedere un listato Do Loop per fare il contrario, cioè riempire l'area col valore della colonna.

```
Sub scrivi_col_dolop()
riga = 12: colonna = 7
Do Until riga = 35
 Do While colonna <> 19
 Cells(riga, colonna).Value = colonna
 colonna = colonna + 1
 Loop
riga = riga + 1: colonna = 7
Loop
End Sub
```

Notiamo subito una netta differenza tra i due cicli, infatti abbiamo dovuto dichiarare i valori iniziali delle variabili riga e colonna prima dell'inizio del ciclo*(riga = 12: colonna = 7)* in questa tipologia ciclica il VBA non riesce a determinare da dove deve iniziare, in quanto la sintassi di espressione è diversa, una volta indicati i valori di partenza troviamo la prima parola chiave *Do Until riga = 35* [ripeti finchè il valore di riga non è uguale a 35] e subito dopo abbiamo posto l'altra condizione che costituisce il ciclo interno *Do While colonna <> 19* facciamo attenzione a questo enunciato, abbiamo usato la parola chiave While e il codice viene interpretato così[ripeti mentre colonna è diversa da 19] ma perchè 19 dato che l'ultima colonna che dobbiamo riempire è la n° [18]?.

Lo comprendiamo subito andando avanti con l'analisi del ciclo, possiamo saltare la spiegazione delle istruzioni *Cells(riga, colonna).Value = colonna* in quanto la loro funzione è uguale a quanto spiegato sopra per il ciclo For con la sola differenza che riempiamo le celle col valore della variabile colonna, mentre la soluzione al quesito posto sta nella riga sotto *colonna = colonna + 1* questo è il nostro contatore, il metodo che usa *Do ... Loop* per incrementarlo e continuare nella sua esecuzione.

Infatti alla prima esecuzione colonna vale 7 e viene incrementata sempre di 1, poi quando trova la parola chiave *Loop* ritorna all'altra parola chiave *Do* (dove ha iniziato il ciclo) ed esegue ancora le istruzioni, ma nella determinante del ciclo abbiamo messo 19 (un valore in più di [18]) semplicemente perchè quando incrementiamo il valore della variabile colonna con la dicitura *colonna = colonna + 1* alla parola chiave *Loop la determinate del ciclo sarebbe soddisfatta se mettessimo [18]* e di conseguenza la nostra area verrebbe riempita fino alla colonna 17

Provate a modificare il valore nell'esempio allegato e comprenderete come agisce il ciclo, inoltre quando si lavora con i cicli e per un motivo qualsiasi ci viene rimandato un errore oppure non viene eseguito quello che volevamo, possiamo usare un piccolo espediente per vedere come agisce il ciclo e quale valore attribuisce alle variabili usate, infatti basta usare la parola chiave Msgbox e possiamo vedere a video il valore della variabile e correggere l'errore. Ecco un esempio di utilizzo

```
Sub scrivi_col_dolop()
riga = 12: colonna = 7
MsgBox riga
Do Until riga = 35
Do While colonna <> 19
MsgBox colonna
Cells(riga, colonna).Value = colonna
colonna = colonna + 1
Loop
riga = riga + 1: colonna = 7
Loop
End Sub
```

L'Istruzione Loop Wend

L'istruzione Loop Wend ripetere un'azione finché una condizione è vera che può essere interpretata come:

While condizione da verificare
azione intrapresa
Wend

Se la condizione è vera, vengono eseguite le azioni indicate nella procedura e quando viene raggiunta l'istruzione Wend, la procedura ritorna all'operazione While e la condizione viene verificata nuovamente. Se la condizione è ancora vera, il processo viene ripetuto, mentre invece se la condizione è falsa, l'esecuzione salta alla prima riga che di codice che segue l'istruzione Wend .

```
Sub Test1()
    Dim i As Integer
    i = 1
    'Loop attraverso le celle della colonna A
     'Si esce dal ciclo se la cella (Cells (i, 1)) è vuota
    While Not IsEmpty(Cells(i, 1))
        'Scrivere il contenuto della cella nella finestra di esecuzione.
        Debug.Print Cells(i, 1)
        'Incrementa la variabile di una unità per testare la cella successiva.
        i = i + 1
    Wend
End Sub
```

L'istruzione Exit Do

È possibile uscire dal ciclo Do While senza completare il ciclo completo, utilizzando la dichiarazione **Exit Do** che arresta immediatamente l'esecuzione del ciclo ed esegue la sezione di codice immediatamente successiva all'istruzione Loop, e nel caso di cicli nidificati si può uscire dal ciclo interno ed eseguire il successivo livello esterno, in quanto si può avere qualsiasi numero di istruzioni Exit Do in un ciclo. Questa istruzione 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, *nTotal* si sommano al valore di 55. Se Range ("A1") contiene il valore 5, il ciclo termina e uscire quando il contatore (es. n) raggiunge i 5 e nTotal si sommano a 10 (si noti che il ciclo non viene eseguito per il controvalore di 5, ed esce il ciclo al raggiungimento di questo valore).

```
Sub Test2 ()
Dim n As Integer , nTotal As Integer
nTotal = 0
Do While n <11
nTotal = n + nTotal
n = n + 1
If n = ActiveSheet.Range ("A1") Then
Exit Do
End If
Loop
MsgBox nTotal
End Sub
```

Forzare l'uscita dal ciclo

Forzare l'uscita di un ciclo si corre sempre il rischio di creare un ciclo infinito se la condizione di uscita non viene mai soddisfatta e in caso di emergenza, se una macro crea un Loop infinito si può fermare la sua esecuzione premendo contemporaneamente i tasti della **Ctrl + Pausa** . È anche possibile premere il tasto Esc e in questo caso viene visualizzata una finestra di errore di esecuzione interrotta, in cui si deve cliccare sul tasto Fine per completare la procedura. Se si desidera gestire i tasti Ctrl + Pausa o Esc in una macro, si deve utilizzare la proprietà *EnableCancelKey*: L'esempio sotto riportato mostra come personalizzare la finestra del messaggio di errore

```
Sub Test3()
    Dim x As Long
    On Error GoTo Fine
    Application.EnableCancelKey = xlErrorHandler
     'si crea un loop per dare tempo di testare la procedura
    'dell'uso del tasto Esc
    For x = 1 To 50000
        Cells(x, 1) = x
    Next x
Fine:
   If Err.Number = 18 Then MsgBox "Operazione Annullata"
End Sub
```