Corso basi di dati
Introduzione al VBScript

Gianluca Di Tomassi
Email: ditomass@dia.uniroma3.it

Università di Roma Tre

IL VBScript

In Microsoft Visual Basic Scripting Edition, il linguaggio di programmazione più recente della famiglia Visual Basic, sono disponibili funzioni per includere script in un'ampia gamma di ambienti diversi, tra cui script client Web in Microsoft Internet Explorer e script server Web in Microsoft Internet Information Server.

VBScript comunica con applicazioni host tramite la tecnologia ActiveX™ Scripting, grazie alla quale nei browser e in altre applicazioni host non è necessario scrivere codice integrativo speciale per ciascun componente script.

Aggiunta di codice VBScript ad una pagina HTML (1)

Il codice VBScript viene inserito all'interno dei tag <SCRIPT>.
Una routine per la verifica di una data di consegna, ad esempio, appare nel modo seguente:

```
<SCRIPT LANGUAGE="VBScript">
<!--
   Function CanDeliver(Dt)
      CanDeliver = (CDate(Dt) - Now()) > 2
   End Function
-->
</SCRIPT>
```

L'attributo LANGUAGE indica il linguaggio di script utilizzato.

Aggiunta di codice VBScript ad una pagina HTML (2)

- Si noti inoltre che la funzione CanDeliver è racchiusa tra i tag di commento, ovvero <!-- e -->, in modo che il codice non venga visualizzato nei browser che non supportano il tag <SCRIPT>.
- È possibile utilizzare i blocchi SCRIPT in qualsiasi punto della pagina HTML, sia nella sezione BODY che nella sezione HEAD.
- Nel caso di form, si consiglia tuttavia di fornire codice script su una stessa riga per rispondere agli eventi di oggetti contenuti nel form. È, ad esempio, possibile incorporare codice script per rispondere alla pressione di un pulsante in un form:

Aggiunta di codice VBScript ad una pagina HTML (3)

```
<HTML>
<HEAD><TITLE>Verifica eventi dei pulsanti</TITLE></HEAD>
<BODY>
<FORM NAME="Form1">
  <INPUT TYPE="Button" NAME="Button1" VALUE="Click">
  <SCRIPT FOR="Button1"EVENT="onClick"
             LANGUAGE="VBScript“ >
      MsgBox "Pulsante premuto"
  </SCRIPT>
</FORM>
</BODY></HTML>
```

Quasi tutto il codice verrà visualizzato nelle routine Sub o Function e verrà richiamato solo quando il codice scritto ne determinerà l'esecuzione

Le routine di VBScript

In VBScript sono disponibili due tipi di routine:

1. le routine Sub
2. le routine Function

- Le routine Sub

Una routine Sub è una serie di istruzioni VBScript incluse tra le istruzioni Sub ed End Sub che eseguono azioni senza restituire alcun valore.

In una routine Sub è possibile specificare argomenti, ovvero costanti, variabili o espressioni passate dalla routine che sta eseguendo una chiamata.

Se non si specifica alcun argomento, nella corrispondente Istruzione SUB è necessario includere parentesi vuote().

ESEMPIO:

```
Sub ConvertTemp()
   Temp = InputBox("Immettere la temperatura in gradi
                    Fahrenheit.", 1)
   MsgBox "La temperatura è pari a " & Celsius(temp) & "
   gradi C."
End Sub
```

In tale routine Sub vengono utilizzate le due funzioni VBScript intrinseche, ovvero incorporate, MsgBox e InputBox, che consentono di richiedere informazioni all'utente.

- Le routine Function

- Una routine Function è una serie di istruzioni VBScript incluse tra le istruzioni Function ed End Function. È simile a una routine Sub, ma a differenza di questa può restituire un valore.
- In una routine Function è possibile specificare argomenti,ovvero costanti, variabili o espressioni passate alla routine dallaroutine chiamante.
- Se non si specifica alcun argomento, nella corrispondente istruzione Function è necessario includere parentesi vuote.

Una routine Function restituisce un valore quando si assegna un valore al nome della routine in una o più istruzioni.

Il tipo restituito di una routine Function è sempre Variant.

ESEMPIO:

```
Sub ConvertTemp()
  temp = InputBox("Immettere la temperatura in gradi
         Fahrenheit.", 1)
  MsgBox "La temperatura è pari a " & Celsius(temp) & "
          gradi C."
End Sub

Function Celsius(fDegrees)
  Celsius = (fDegrees - 32) * 5 / 9
End Function
```


- la funzione Celsius consente di eseguire la conversione da Fahrenheit a Celsius.
- Quando la funzione viene richiamata nella routine Sub convertTemp, viene passata alla funzione una variabile contenente il valore dell'argomento.
- Il risultato del calcolo viene quindi restituito alla routine che ha eseguito la chiamata e visualizzato in una finestra di messaggio.

Come includere e leggere dati dalle routine

- I dati vengono passati alla routine tramite **argomenti** che fungono da segnaposto per i dati stessi.
- Agli argomenti è possibile assegnare un nome valido per le variabili.
- Quando si crea una routine utilizzando l'istruzione **Sub** o **Function**, è necessario far seguire il nome della routine da parentesi, all'interno delle quali vengono inseriti gli argomenti separati da una virgola.

ESEMPIO:

```
Function Celsius(fDegrees)
   Celsius = (fDegrees-32)*5/9
End Function
```

`fDegrees` è il segnaposto del valore da convertire passato alla funzione Celsius

IMPORTANTE:

Per ottenere dati da una routine, è necessario utilizzare routine Function che, a differenza delle routine Sub, possono restituire un valore.

NOTA:

Per utilizzare routine Function nel codice, è necessario includerle a destra delle assegnazioni di variabili oppure in un'espressione.

ESEMPIO:

```
Temp = Celsius(fDegrees)
MsgBox "La temperatura in gradi Celsius è pari a " & Temp & "
        gradi.“
```

Per richiamare una routine Sub da un'altra routine, è sufficiente specificarne il nome insieme ai valori degli argomenti obbligatori, separati da una virgola. Può in alternativa essere utilizzata l'istruzione Call che non è obbligatoria. Se viene utilizzata, è necessario racchiudere gli eventuali argomenti tra parentesi.

ESEMPIO:

```
Call MyProc(primoarg,secondoarg)
MyProc primoarg, secondoarg
```

Vengono eseguite due chiamate alla routine myProc. Sebbene l'istruzione Call sia utilizzata nel codice di una sola chiamata, entrambe le chiamate eseguono la stessa operazione.

Tipi di dati in VBScript

In VBScript è disponibile solo il tipo di dati Variant, un tipo di dati speciale che, a seconda della modalità in cui viene utilizzato, può includere vari tipi di informazioni.

Il tipo di dati Variant, essendo l'unico disponibile, è il tipo di dati restituito da tutte le funzioni di VBScript.

Nella forma più semplice una variabile Variant può includere informazioni numeriche o stringhe. È equivalente a un numero se utilizzata in un contesto numerico e a una stringa se utilizzata in un contesto stringa.

È tuttavia possibile fare in modo che i numeri vengano gestiti come stringhe racchiudendoli tra virgolette doppie (" ").

Sottotipi (1)

È tuttavia possibile utilizzare informazioni numeriche di vario tipo con dimensioni che vanno dai valori booleani ai numeri in virgola mobile di grandi dimensioni. Le varie categorie di informazioni che possono essere incluse in variabili Variant sono definite sottotipi.

Sottotipi (2)

Sottotipo	Descrizione
Empty	Variabile Variant non inizializzata. Il valore è 0 nel caso di variabili numeriche e una stringa di lunghezza zero ("") nel caso di variabili stringa.
Null	Variabile Variant che include dati non validi inseriti volutamente
Boolean	Può contenere True o False.
Byte	Contiene un intero compreso tra 0 e 255.
Integer	Contiene un intero compreso tra -32.768 e 32.767.
Currency	Contiene un valore compreso tra -922.337.203.685.477
Long	Contiene un intero compreso tra -2.147.483.648 e 2.147.483.647.

Sottotipi (3)

Sottotipo	Descrizione
Single	Contiene un numero in virgola mobile e precisione singola compreso tra -3 e 3
Double	Contiene un numero in virgola mobile e precisione doppia compreso tra -1 e 1
Date (Time)	Contiene un numero che rappresenta una data compresa tra l'1 gennaio dell'anno 100 e il 31 dicembre del 9999.
String	Contiene una stringa di lunghezza variabile composta da un massimo di circa 2 miliardi di caratteri.
Object	Contiene un oggetto.
Error	Contiene un numero di errore.

Definizione di una variabile

Una variabile è un segnaposto che fa riferimento a una posizione della memoria del computer e in cui è possibile memorizzare informazioni di programma soggette a modifiche durante l'esecuzione degli script.

È possibile, ad esempio, creare la variabile ClickCount per memorizzare quante volte l'utente fa clic su un oggetto di una pagina Web specifica.

Per visualizzarne o modificarne il valore, è sufficiente farvi riferimento in base al nome.

In VBScript, le variabili sono **sempre** di tipo Variant.

Le variabili vengono dichiarate in modo esplicito nello script utilizzando le istruzioni Dim, Public e Private.

Esempio: `Dim DegreesFahrenheit`

Per dichiarare più variabili, è necessario separare i vari nomi con una virgola.

Esempio: `Dim Top, Bottom, Left, Right`

Restrizioni relative ai nomi

- Il nome di una variabile:
- Deve iniziare con una lettera dell'alfabeto.
- Non può includere punti.
- Non deve essere composto da più di 255 caratteri.
- Deve essere univoco nell'area di validità in cui la variabile è dichiarata.

Area di validità e vita utile delle variabili

L'area di validità di una variabile varia a seconda della posizione in cui la variabile viene dichiarata.

1. Quando si dichiara una variabile in una routine, è possibile accedere o modificare il corrispondente valore solo tramite il codice di tale routine.
   ⇒ Si tratta di una variabile a livello di routine con area di validità locale.

2. Se una variabile viene dichiarata all'esterno di una routine, essa sarà riconosciuta in tutte le altre routine dello script.
   ⇒ variabile definita a livello di script, è associata un'area di validità a livello di script.

Vita utile delle variabili

La vita utile di una variabile corrisponde alla sua durata.

Per una ***variabile a livello di script*** la vita utile è compresa tra la dichiarazione e la conclusione dell'esecuzione dello script.

Per una ***variabile a livello di routine*** le variabili esistono solo mentre è in esecuzione la routine in cui sono dichiarate.

Al termine dell'esecuzione della routine la variabile viene eliminata.

NOTA: Le variabili locali rappresentano uno spazio di memorizzazione ideale quando si esegue una routine. In più routine è possibile includere variabili locali aventi lo stesso nome in quanto ciascuna di esse viene riconosciuta solo dalla routine in cui è dichiarata.

Assegnazione di valori alle variabili

Per assegnare valori alle variabili, è necessario creare un'espressione indicando a sinistra la variabile e a destra il valore che si desidera assegnare.

Esempio: `B = 200`

Variabili scalari e variabili matrice

Distinguiamo tra:

- Una variabile contenente un solo valore è definita variabile ***scalare***.
- Una variabile contenente una serie di valori è definita variabile ***matrice***.

La dichiarazione di queste variabili è simile a quella delle variabili scalari, con la sola differenza che, nelle dichiarazioni di variabili matrici, il nome della variabile è seguito da parentesi.

Esempio:

`Dim A(10)` dichiarazione di una matrice a una sola dimensione contenente 11 elementi:

ATTENZIONE

Anche se tra parentesi è indicato il numero 10, la matrice include in effetti 11 elementi.

In VBScript infatti le matrici sono sempre in base zero e di conseguenza il numero degli elementi in esse contenuti corrisponde sempre al numero indicato tra parentesi più uno.

Queste **matrici** sono definite a **dimensione fissa**.

Per assegnare dati a ciascun elemento di una matrice, è necessario utilizzare un indice.

Esempio:

```
A(0) = 256
A(1) = 324
A(2) = 100
. . .
A(10) = 55
```


Per recuperare dati da un elemento della matrice, è possibile procedere nello stesso modo, ovvero inserire un indice nell'elemento desiderato.

Esempio:

```
SomeVariable = A(8)
```

- Le matrici non sono limitate a una sola dimensione.
- È possibile specificare fino a 60 dimensioni.
- Per dichiarare dimensioni multiple, è necessario separare con una virgola i valori delle dimensioni indicati tra parentesi.

Esempio:

```
Dim MyTable(5, 10)
```

la variabile MyTable è una matrice a due dimensioni composta da 6 righe e 11 colonne.

NOTA:

In una matrice a due dimensioni il primo numero corrisponde sempre al numero di righe, il secondo al numero di colonne.

È possibile dichiarare matrici le cui dimensioni vengono modificate durante l'esecuzione dello script.

Questo tipo di matrici sono definite matrici dinamiche e vengono inizialmente dichiarate in una routine utilizzando l'istruzione Dim come qualsiasi altra matrice, oppure l'istruzione ReDim.

DIFFERENZA TRA MATRICI A DIMENSIONE FISSA E VARIABILI:
La differenza è che tra parentesi non viene indicata alcuna grandezza o numero di dimensioni. Ad esempio:

Esempio:

```
Dim MyArray()
ReDim AnotherArray()
```

Per utilizzare una matrice dinamica, è necessario specificare più istruzioni ReDim in modo da determinare il numero di dimensioni e la grandezza di ciascuna dimensione.

Esempio:

```
ReDim MyArray(25)
```

l'istruzione ReDim imposta la grandezza iniziale della matrice dinamica su 25.

```
ReDim Preserve MyArray(30)
```

Una successiva istruzione ReDimridimensiona quindi la matrice impostandola su 30 e viene utilizzata la parola chiave **Preserve** per mantenere il contenuto della matrice durante l'operazione di ridimensionamento

NOTA:

Non esiste alcun limite alla frequenza delle operazioni di ridimensionamento delle matrici dinamiche.

***RICORDARE*:**

Se si riducono le dimensioni di una matrice, i dati contenuti negli elementi eliminati vanno perduti.

Costanti in VBScript

Una costante è un nome significativo non soggetto a modifiche utilizzato in sostituzione di un numero o di una stringa.

Per creare costanti in VBScript, è necessario utilizzare l'istruzione Const con cui vengono create costanti stringa o numeriche con nomi significativi

Esempio:

```
Const MyString = "Questa è la mia stringa."
Const MyAge = 49
```

NOTA:

Il valore letterale stringa è racchiuso tra virgolette doppie ("), il modo più chiaro per contraddistinguere i valori stringa dai valori numerici

È possibile rappresentare i valori letterali di data e ora racchiudendoli tra simboli di cancelletto (#)

Esempio:

```
Const CutoffDate = #6-1-97#
```

CONSIGLIO:

Adottare una convenzione di denominazione in modo da contraddistinguere le costanti dalle variabili ed evitare quindi di riassegnare valori alle costanti durante l'esecuzione dello script. Ad esempio si può utilizzare il prefisso "vb" o "con" per i nomi di costanti o assegnare nomi in maiuscolo.

Utilizzo di istruzioni condizionali

È possibile controllare il flusso di script tramite istruzioni condizionali e istruzioni di ciclo. Con le istruzioni condizionali è possibile scrivere codice VBScript che consente di prendere decisioni e ripetere azioni.

Istruzioni condizionali disponibili:

- If…Then…Else
- Select Case

L'istruzione IF..Then..Else

Consente di valutare se una condizione è True o False e, a seconda del risultato, di specificare una o più istruzioni da eseguire.

In genere la condizione corrisponde a un'espressione in cui due valori o due variabili vengono confrontati tramite un operatore di confronto.

Le istruzioni *If…Then…Else* possono essere nidificate in un numero qualsiasi di livelli di nidificazione.

Per eseguire una sola istruzione quando una condizione è **True**, è necessario utilizzare la sintassi a riga singola dell'istruzione **If…Then…Else**

Esempio:

```
Sub FixDate()
   Dim myDate
   myDate = #2/13/95#
   If myDate < Now Then myDate = Now
End Sub
```

Per eseguire più righe di codice, è necessario utilizzare la sintassi a righe multiple o a blocco, in cui è inclusa l'istruzione End If

ESEMPIO:

```
Sub AlertUser(value)
 If value = 0 Then
   AlertLabel.ForeColor = vbRed
   AlertLabel.Font.Bold = True
   AlertLabel.Font.Italic = True
 End If
End Sub
```


È possibile utilizzare un'istruzione If...Then...Else per definire due blocchi di istruzioni da eseguire l'uno quando la condizione è True e l'altro quando la condizione è False.

ESEMPIO:

```
Sub AlertUser(value)
      If value = 0 Then
            AlertLabel.ForeColor = vbRed
            AlertLabel.Font.Bold = True
            AlertLabel.Font.Italic = True
      Else
            AlertLabel.Forecolor = vbBlack
            AlertLabel.Font.Bold = False
            AlertLabel.Font.Italic = False
      End If
End Sub
```

Aggiungendo nell'istruzione If...Then...Else proposizioni ElseIf, è possibile scegliere tra situazioni diverse. La funzionalità dell'istruzione If...Then...Else viene in tal modo estesa consentendo di controllare il flusso del programma in base a possibilità diverse

ESEMPIO:

```
Sub ReportValue(value)
    If value = 0 Then
                        MsgBox value
    ElseIf value = 1 Then
                        MsgBox value
    ElseIf value = 2 Then
                        Msgbox value
    Else
          Msgbox "Valore non compreso …"
    End If
End Sub
```


L'istruzione "Select Case"

Per fornire più situazioni diverse, è possibile aggiungere il numero necessario di proposizioni ElseIf.

Un utilizzo eccessivo di queste proposizioni può tuttavia creare un codice poco chiaro e contorto.

Un metodo alternativo migliore consiste nell'utilizzare l'istruzione **Select Case**.

La struttura Select Case rappresenta un'alternativa all'istruzione **If…Then…ElseIf** per la selezione di un blocco di istruzioni specifico tra più blocchi diversi.

Lettura di una “Select Case”:

NOTA: La funzione dell'istruzione Select Case è simile a quella dell'istruzione If...Then...Else, con la differenza tuttavia che il codice risulta più efficiente e di più facile lettura.

1. Una struttura Select Case include una singola espressione di testo valutata una sola volta all'inizio della struttura stessa.

2. Il risultato dell'espressione viene quindi confrontato con i valori di ciascun blocco Case della struttura.

3. Se viene individuata una corrispondenza, il blocco di istruzioni associato al blocco Case specifico verrà eseguito.

Differenza rispetto ad una If…Then…Else

A differenza della struttura Select Case con cui un'espressione viene valutata una sola volta all'inizio della struttura stessa, la struttura If...Then...ElseIf consente di valutare un'espressione diversa per ciascuna istruzione ElseIf. È possibile sostituire una struttura If...Then...ElseIf con una struttura Select Case solo se con ciascuna istruzione ElseIf viene valutata la stessa espressione.

ESEMPIO:

```
Select Case tipocarta
Case "MasterCard"
   DisplayMCLogo
   ValidateMCAccount
Case "Visa"
   DisplayVisaLogo
   ValidateVisaAccount
Case "American Express"
   DisplayAMEXCOLogo
   ValidateAMEXCOAccount
Case Else
   DisplayUnknownImage
   PromptAgain
End Select
```


Utilizzo di cicli all'interno del codice

L'esecuzione di cicli consente di eseguire ripetutamente un gruppo di istruzioni.

In alcuni cicli le istruzioni vengono ripetute fino a quando una condizione risulta False, mentre in altri vengono ripetute fino a quando una condizione risulta True.

Esistono inoltre cicli in cui le istruzioni vengono ripetute un numero specifico di volte.

Cicli disponibili

In VBScript sono disponibili le seguenti istruzioni per l'esecuzione di cicli:

Do…Loop: le istruzioni vengono ripetute fino a quando una condizione risulta True.

While…Wend: le istruzioni vengono ripetute fino a quando una condizione risulta True.

For…Next: le istruzioni vengono ripetute il numero di volte specificato da un contatore.

For Each…Next: un gruppo di istruzioni viene ripetuto per ciascun elemento incluso nell'insieme di una matrice.

Ciclo “Do… Loop”

È possibile utilizzare le istruzioni **Do…Loop** per eseguire un blocco di istruzioni un numero indefinito di volte.

Le istruzioni vengono ripetute mentre una condizione è True oppure fino a quando risulta True.

Per verificare una condizione in un'istruzione **Do…Loop**, è necessario utilizzare la parola chiave **While**.

La verifica può essere eseguita prima dell'inizio del ciclo, oppure dopo almeno una esecuzione del ciclo.

ESEMPIO di verifica eseguita prima dell'inizio del ciclo

```
Sub ChkFirstWhile()
  Dim counter, myNum
  counter = 0
  myNum = 20
  Do While myNum > 10
    myNum = myNum - 1
    counter = counter + 1
  Loop
  MsgBox "Il ciclo ha eseguito " & counter & "
  ripetizioni."
End Sub
```

In tale routine se myNum è impostato su 9 anziché su 20, le istruzioni incluse nel ciclo non verranno mai eseguite

ESEMPIO di verifica eseguita dopo almeno una esecuzione del ciclo

```
Sub ChkLastWhile()
  Dim counter, myNum
  counter = 0
  myNum = 9
  Do
       myNum = myNum - 1
       counter = counter + 1
  Loop While myNum > 10
  MsgBox "Il ciclo ha eseguito " & counter & "
  ripetizioni."
End Sub
```

In tale routine, le istruzioni incluse nel ciclo vengono eseguite una sola volta, in quanto la condizione è già False.

Per verificare una condizione in un'istruzione Do...Loop, è possibile utilizzare la parola chiave **Until** in due modi diversi:

1.la verifica può essere eseguita prima dell'inizio del ciclo;

2.la verifica può essere eseguita dopo almeno una esecuzione del ciclo.

ESEMPIO di verifica eseguita prima dell'inizio del ciclo

```
Sub ChkFirstUntil()
    Dim counter, myNum
    counter = 0
    myNum = 20
    Do Until myNum = 10
          myNum = myNum - 1
          counter = counter + 1
    Loop
    MsgBox "Il ciclo ha eseguito " & counter & "
    ripetizioni."
End Sub
```

ESEMPIO di verifica eseguita dopo almeno una esecuzione del ciclo

```
Sub ChkLastUntil()
  Dim counter, myNum
  counter = 0
  myNum = 1
  Do
     myNum = myNum + 1
     counter = counter + 1
  Loop Until myNum = 10
  MsgBox "Il ciclo ha eseguito " & counter & "
  ripetizioni."
End Sub
```

Il ciclo viene ripetuto finché la condizione risulta False

Forzare l'uscita da un ciclo

Per uscire da una struttura Do...Loop, è possibile utilizzare l'istruzione **Exit Do** (utile per evitare la ripetizione infinita di un ciclo).

L'istruzione Exit Do deve essere inclusa nel blocco di istruzioni True di un'istruzione If...Then...Else. Se la condizione è False, il ciclo viene eseguito normalmente.

Esempio: Cosa succede in questa funzione ?

```
Sub ExitExample()
  Dim counter, myNum
  counter = 0
  myNum = 9
  Do Until myNum = 10
   myNum = myNum - 1
   counter = counter + 1
   If myNum < 10 Then Exit Do
  Loop
MsgBox "Il ciclo ha eseguito " & counter & " ripetizioni."
End Sub
```

SPIEGAZIONE:

a myNum viene assegnato un valore in base al quale viene creato un ciclo infinito. L'istruzione If...Then...Else consente di verificare questa condizione, impedendo pertanto un'esecuzione all'infinito.

Ciclo “While… Wend”

La sintassi è la seguente:

```
While condizione
    [istruzioni]
Wend
```

Se l'argomento *condizione* è **True**, tutte le istruzioni specificate nell'argomento *istruzioni* verranno eseguite fino a quando non viene individuata l'istruzione **Wend**. Il controllo ritorna quindi all'istruzione **While** e la *condizione* viene nuovamente verificata.

Esempio:

```
Dim Counter Counter = 0    ' Inizializza la variabile.
While Counter < 20         ' Verifica il valore di Counter.
   Counter = Counter + 1   ' Incrementa Counter.
   Alert Counter
Wend              ' Interrompe il ciclo While quando Counter > 19.
```

Ciclo “For… Next”

L'istruzione **For…Next** consente di eseguire un blocco di istruzioni un numero specifico di volte.

Nel caso di cicli, è necessario utilizzare una variabile contatore il cui valore viene aumentato o diminuito a ogni ripetizione del ciclo.

Esempio:

```
Sub DoMyProc50Times()
  Dim x
  For x = 1 To 50
      MyProc
  Next
End Sub
```

La routine MyProc viene eseguita 50 volte. L'istruzione **For** specifica la variabile contatore x e il corrispondente valore iniziale e finale.

L'istruzione **Next** consente di aumentare la variabile contatore con incrementi di una unità.

La parola chiave **Step** consente di aumentare o diminuire la variabile contatore del valore specificato.

ESEMPIO:

```
Sub TwosTotal()
  Dim j, total
  For j = 2 To 10 Step 2
     total = total + j
  Next
  MsgBox "Il totale è " & total
End Sub
```

La variabile contatore viene incrementata di due unità a ogni ripetizione del ciclo

DOMANDA: Quale è il totale alla fine del ciclo ?

RISPOSTA: Il totale corrisponde alla somma di 2, 4, 6, 8, 10

Per diminuire la variabile contatore, è necessario utilizzare un valore **Step** negativo specificando un valore finale minore del valore iniziale.

ESEMPIO:

```
Sub NewTotal()
  Dim myNum, total
  For myNum = 16 To 2 Step -2
      total = total + myNum
  Next
  MsgBox "Il totale è " & total
End Sub
```

La variabile contatore `myNum` viene diminuita di due unità a ogni ripetizione del ciclo

DOMANDA: Quale è il totale alla fine del ciclo ?

RISPOSTA: Il totale corrisponde alla somma di 16, 14, 12, 10, 8, 6, 4 e 2.

Per uscire da un'istruzione **For…Next** prima che il valore del contatore abbia raggiunto il valore finale, è necessario utilizzare l'istruzione **Exit For**.

NOTA: Dato che l'uscita da un'istruzione **For…Next** viene in genere eseguita solo in determinate situazioni, ad esempio quando si verifica un errore, l'istruzione **Exit For** deve essere inclusa nel blocco di istruzioni **True** di un'istruzione **If…Then…Else**. Se la condizione è False, il ciclo viene eseguito normalmente

Ciclo "For each… Next"

Tale ciclo è simile a **For…Next**, con la sola differenza che le istruzioni non vengono ripetute il numero di volte specificato, ma per ciascun elemento di un insieme di oggetti o per ciascun elemento di una matrice

DOMANDA: Quale è l'utilità di tale ciclo ?

RISPOSTA: Risulta particolarmente utile quando non si conosce il numero di elementi di un insieme.

ESEMPIO 1

```
<HTML> <HEAD>
<TITLE>Una semplice pagina di esempio</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!-- Sub Button1_OnClick
           MsgBox "Mirabile visu."
      End  Sub -->
</SCRIPT> </HEAD>
<BODY>
<H3>Una semplice pagina di esempio</H3><HR>
<FORM>
  <INPUT NAME="Button1" TYPE="BUTTON" VALUE="Fare clic qui">
</FORM>
</BODY>
</HTML>
```


ESEMPIO 2

```
<HTML>
<HEAD><TITLE>Convalida semplice</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Submit_OnClick
  Dim TheForm
  Set TheForm = Document.ValidForm
  If IsNumeric(TheForm.Text1.Value) Then
  If TheForm.Text1.Value < 1 Or TheForm.Text1.Value >10 Then
      MsgBox "Immettere un numero compreso tra 1 e 10."
  Else
      MsgBox "Valore corretto."
    End If
  Else
    MsgBox "Immettere un valore numerico."
  End If
End Sub
```

```
-->
</SCRIPT>
</HEAD>
<BODY>
<H3>Convalida semplice</H3><HR>
<FORM NAME="ValidForm">
Immettere un valore compreso tra 1 e 10:
<INPUT NAME="Text1" TYPE="TEXT" SIZE="2">
<INPUT NAME="Submit" TYPE="BUTTON" VALUE="Invia">
</FORM>
</BODY>
</HTML>
```