Le Selezioni condizionali

Se si dispone di un gran numero di condizioni da esaminare, la dichiarazione *If ... Then ... Else* è costretta a verificarle tutte, ma il linguaggio Visual Basic offre l'alternativa di saltare direttamente all'istruzione che si applica allo stato di una condizione. Questo processo è indicato come selezionare le condizioni utilizzando le parole chiave **Select Case** e **Case**, la cui formula è:

```
Select Case Expression
    Case Expression1
        Statement1
    Case Expression2
        Statement2
    Case ExpressionX
        StatementX
End Select
```

La dichiarazione inizia con **Select Case** e termina con **End Select** e sul lato destro del Select Case, si deve immettere il valore, *Expression*, che verrà utilizzato, che può essere un valore booleano, un carattere o una stringa, un numero naturale, un numero decimale, un valore di data o ora, oppure una enumerazione.

All'interno del *Select Case* e prima della linea *End Select*, è necessario fornire una o più sezioni e ognuna contiene una parola chiave **Case** seguita da un valore sul lato destro e i valori di *Expresion1*, *Expresion2* o *ExpresionX*, devono essere dello stesso tipo come il valore di Expression. Dopo il Case e la sua espressione (Expression), si può scrivere una dichiarazione e quando si accede a questa sezione di codice, viene considerato il valore di Expression e viene confrontato con ciascun valore di Expression di ogni Case:

- Se il valore di Expression1 è uguale a quella di Expression, allora statement1 viene eseguito.
- Se il valore di Expression1 non è uguale a quella di Expression, l'interprete si sposta Expression2
- Se il valore di Expression2 è uguale a quella di Expression, allora viene eseguita l'istruzione Statement2
- Questo continuerà fino all'ultimo ExpressionX

Ecco un esempio:

```
Sub Test()
   Dim risp As Byte

   risp = CByte(InputBox( _
              "Una delle seguenti parole non è una parola chiave di Visual Basic" & vbCrLf &
vbCrLf & _
                "1) Function" & vbCrLf & _
                "2) Except" & vbCrLf & _
                "3) ByRef" & vbCrLf & _
                "4) Each" & vbCrLf & vbCrLf & _
                "Inserisci la tua risposta "))

   Select Case risp
      Case 1
         MsgBox ("Giusto: Function è una parola chiave di Visual Basic " & vbCrLf & _
               "Viene utilizzata per creare una procedura di tipo funzione")
      Case 2
         MsgBox ("Sbagliato: Except non è una parola chiave di " & vbCrLf & _
               "Visual Basic ma è una parola chiave " & vbCrLf & _
               "utilizzata in C++ per la Gestione delle eccezioni")
      Case 3
         MsgBox ("Giusto: ByRef è una parola chiave di Visual Basic" & vbCrLf & _
               "Viene utilizzata per passare un argomento a una procedura")
      Case 4
         MsgBox ("Giusto: Each è una parola chiave di Visual Basic " & vbCrLf & _
               "utilizzata in un ciclo per scorrere una lista di voci ")
   End Select
End Sub
```

Ecco un esempio di esecuzione del programma:

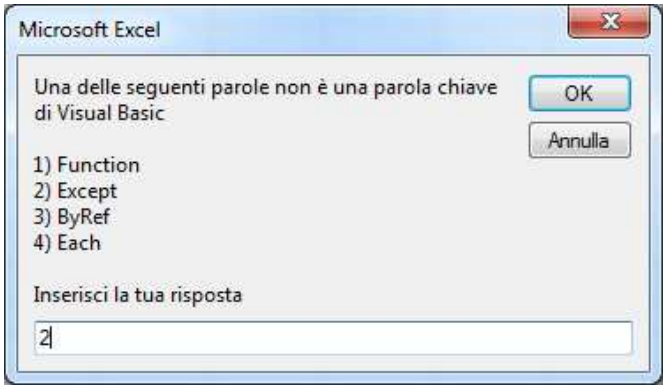

Fig. 1

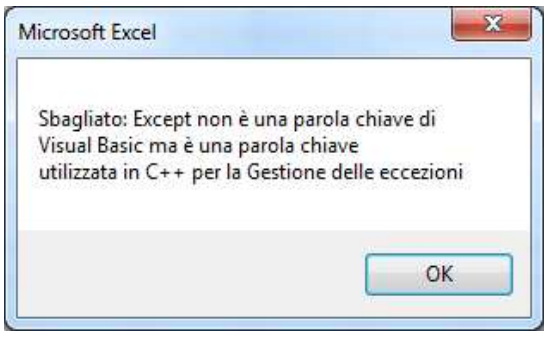

Fig. 2

Il codice di cui sopra presuppone che uno dei Case corrisponderà al valore di Expression, ma non è sempre così. Se si prevede che ci potrebbe essere nessuna corrispondenza tra l'espressione (Expression) e uno dei vari Expression dei Case, è possibile utilizzare una dichiarazione Case Else alla fine della lista. La dichiarazione sarebbe allora simile a questa:

```
Select Case Expression
   Case Expression1
      Statement1
   Case Expression2
      Statement2
   Case Expression3
      Statement3
   Case Else
      Statementk
End Select
```

In questo caso, la dichiarazione dopo *Case Else* viene eseguita se nessuna delle espressioni precedenti corrisponde alla espressione principale. Ecco un esempio:

```
Sub Test()
   Dim risp As Byte

   risp = CByte(InputBox( _
              "Una delle seguenti parole non è una parola chiave di Visual Basic" & vbCrLf &
vbCrLf & _
                 "1) Function" & vbCrLf & _
                 "2) Except" & vbCrLf & _
                 "3) ByRef" & vbCrLf & _
                 "4) Each" & vbCrLf & vbCrLf & _
                 "Inserisci la tua risposta "))

   Select Case risp
      Case 1
         MsgBox ("Giusto: Function è una parola chiave di Visual Basic " & vbCrLf & _
               "Viene utilizzata per creare una procedura di tipo funzione")
      Case 2
         MsgBox ("Sbagliato: Except non è una parola chiave di " & vbCrLf & _
               "Visual Basic ma è una parola chiave " & vbCrLf & _
               "utilizzata in C++ per la Gestione delle eccezioni")
      Case 3
         MsgBox ("Giusto: ByRef è una parola chiave di Visual Basic" & vbCrLf & _
               "Viene utilizzata per passare un argomento a una procedura")
      Case 4
         MsgBox ("Giusto: Each è una parola chiave di Visual Basic " & vbCrLf & _
               "utilizzata in un ciclo per scorrere una lista di voci ")
      Case Else
         MsgBox ("Selezione non valida")
   End Select
End Sub
```

Ecco un esempio di esecuzione del programma:

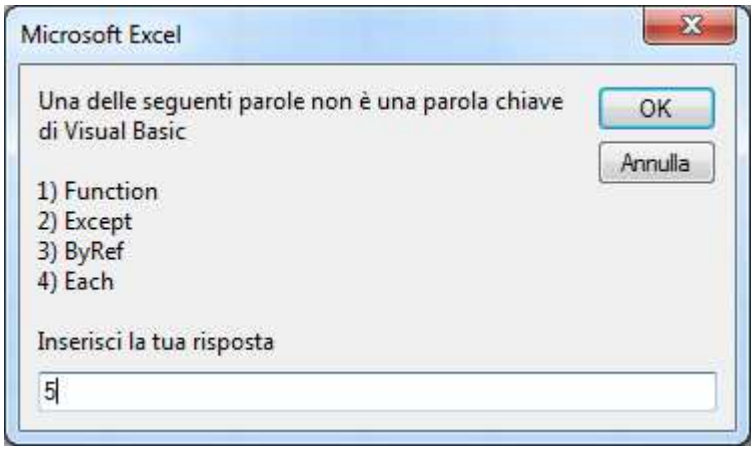

Fig. 3

Fig. 4

Come accennato nell'introduzione, il Select Case può utilizzare un valore diverso da un numero intero, ad esempio, è possibile utilizzare un carattere:

```
Sub Test()
    Dim sesso As String
    sesso = "M"

    Select Case sesso
        Case "F"
            MsgBox ("Femmina")
        Case "M"
            MsgBox ("Maschio")
        Case Else
            MsgBox ("Sconosciuto")
    End Select
    End Sub
```

Questo produrrebbe:

Fig. 5

Si noti che in questo caso stiamo usando solo caratteri maiuscoli, se si desidera convalidare anche caratteri minuscoli, potremmo essere costretti a creare sezioni aggiuntive di Case per ciascuno. Ecco un esempio:

```
Sub Test()
   Dim sesso As String
   sesso = "f"

   Select Case sesso
      Case "F"
         MsgBox ("Femmina")
      Case "f"
         MsgBox ("Femmina")
      Case "M"
         MsgBox ("Maschio")
      Case "m"
         MsgBox ("Maschio")
      Case Else
         MsgBox ("Sconosciuto")
   End Select
   End Sub
```

Questo produrrebbe:

Fig. 6

Invece di utilizzare un valore per Case, è possibile applicare più di una condizione, per fare ciò, sul lato destro della parola chiave Case, è possibile separare le espressioni con una virgola. Ecco alcuni esempi:

```
Sub Test()
   Dim sesso As String
   sesso = "F"

   Select Case sesso
      Case "f", "F"
         MsgBox ("Femmina")
      Case "m", "M"
         MsgBox ("Maschio")
      Case Else
         MsgBox ("Sconosciuto")
   End Select
   End Sub
```

Convalida di una serie di casi

È possibile utilizzare un intervallo di valori per un Case, a tale scopo, sul lato destro del Case, si deve inserire il valore più basso, seguito dalla parola chiave To e seguito dal valore più alto. Ecco un esempio:

```
Sub Test()
    Dim eta As Integer
    eta = 24
    Select Case eta
        Case 0 To 17
            MsgBox ("Ragazzo")
        Case 18 To 55
            MsgBox ("Adulto")
        Case Else
            MsgBox ("Anziano")
    End Select
End Sub
```

Questo produrrebbe:

Fig. 7

Verificare un valore

Si consideri la seguente procedura:

```
Sub Test()
    Dim num As Integer
    num = 448

    Select Case num
        Case -602
            MsgBox ("-602")
        Case 24
            MsgBox ("24")
        Case 0
            MsgBox ("0")
    End Select
End Sub
```

Ovviamente questa Select Case lavorerà in rari casi solo quando l'espressione di un Case corrisponde esattamente al valore richiesto, in realtà, per questo tipo di scenario, è possibile convalidare un intervallo di valori.

Il linguaggio Visual Basic fornisce un'alternativa, infatti è possibile controllare se il valore della espressione risponde a un criterio anziché a un valore esatto. Per crearla, si utilizza l'operatore **Is** con la seguente formula:

Is Operator Value

Si inizia con la parola chiave Is seguita da uno degli operatori booleani che abbiamo visto precedentemente, come: =, <>, <, <=,> o> = e sul lato destro dell'operatore booleano, si digita il valore desiderato. Ecco alcuni esempi:

```
Sub Test()
   Dim num As Integer
   num = -448

   Select Case num
      Case Is < 0
         MsgBox ("Il numero è negativo")
      Case Is > 0
         MsgBox ("Il numero è positivo")
      Case Else
         MsgBox ("Il numero è 0")
   End Select
End Sub
```

Anche se in questo esempio abbiamo usato un numero naturale, è possibile utilizzare qualsiasi confronto logico appropriato in grado di produrre un risultato vero o un falso, inoltre è possibile combinare con le altre alternative che abbiamo visto in precedenza, come ad esempio separando le espressioni di un Case con una virgola. Con la dichiarazione Select Case, abbiamo visto come controllare valori diversi e di intervenire. Ecco un esempio:

```
Sub Test()
   Dim num As Integer, cat As String
   num = 2

   Select Case num
      Case 1
         cat = "Ragazzo"
      Case 2
         cat = "Adulto"
      Case Else
         cat = "Anziano"
   End Select

   MsgBox ("La categoria è : " & cat)
End Sub
```

Questo produrrebbe:

Fig. 8

Abbiamo anche visto che il linguaggio Visual Basic fornisce la funzione **Choose** che può controllare una condizione e intraprendere un'azione, inoltre è un'altra alternativa a una dichiarazione Select Case. Se prendiamo in considerazione la sintassi della funzione Choose:

```
Function Choose( ByVal Index As Double,  ByVal ParamArray Choice() As Variant ) As Object
```

Si può notare che questa funzione richiede due argomenti, il primo argomento è equivalente a *Expression* della nostra Select Case, e come già detto, il primo argomento deve essere un numero che sarà il valore centrale rispetto al quale saranno confrontati gli altri valori. Invece di utilizzare sezioni Case si forniscono i corrispondenti valori di *ExpressionX* come un elenco di valori al posto del secondo argomento, separandoli da virgole. Ecco un esempio:

```
Choose(num, "Ragazzo", "Adulto", "Anziano")
```

Come già menzionato, i valori del secondo argomento sono forniti come una lista e ogni membro della lista utilizza un indice. Il primo membro della lista, che è il secondo argomento di questa funzione, ha un indice pari a 1, il secondo valore dell'argomento, che è il terzo argomento della funzione, ha un indice di 2. È possibile continuare ad aggiungere i valori del secondo argomento come meglio si crede. Quando la funzione Choose è stata richiamata, restituisce un valore di tipo Variant ed è possibile recuperare tale valore, memorizzandolo in una variabile e usarlo quando se ne ha la necessità. Ecco un esempio:

```
Sub Test()
    Dim num As Integer, cat As String
    num = 1
    cat = Choose(num, "Ragazzo", "Adulto", "Anziano")
    MsgBox ("Il tipo di categoria è: " & cat)
End Sub
```

Questo produrrebbe:

Fig. 9

Fino ad ora, abbiamo visto come creare istruzioni condizionali normali. Ecco un esempio:

```
Sub Test()
   Dim num%
   num% = InputBox("Inserisci un numero inferiore a 5")

   If num% >= 0 Then
      MsgBox (num%)
   End If
End Sub
```

Quando questa procedura viene eseguita, l'utente è invitato a fornire un numero, se questo numero è positivo, verrà visualizzata una finestra di messaggio, mentre se l'utente inserisce un numero negativo, non succede nulla. In un tipico programma, dopo la convalida di una condizione, si consiglia di agire e per fare questo, è possibile creare una sezione del programma all'interno della istruzione condizionale di convalida, ma in realtà, è possibile creare un'istruzione condizionale all'interno di un'altra istruzione condizionale. Questo è indicato come nidificazione condizione. Qualsiasi condizione può essere nidificato all'interno di un'altra e possono essere incluse più condizioni all'interno di un'altra. Ecco un esempio in cui una condizione If nidificata all'interno di un altro If:

```
Sub Test()
   Dim num%
   num% = InputBox("Inserisci un numero inferiore a 5")

   If num% >= 0 Then
      If num% < 12 Then
      MsgBox (num%)
   End If
   End If
End Sub
```

L'istruzione Goto

L'istruzione **Goto** consente di passare ad un'altra sezione del programma mentre è in esecuzione e per poter utilizzare l'istruzione Goto, si deve inserire un nome su una particolare sezione del procedimento in modo da poter fare riferimento a tale nome. Il nome, chiamato anche etichetta, è fatto di una sola parola e segue le regole che abbiamo applicato ai nomi e poi seguito da due punti ":". Ecco un esempio:

```
Sub Test()
Etichetta1:
End Sub
```

Dopo aver creato l'etichetta, è possibile elaborarla, inserendo nel codice l'istruzione GoTo seguita dall'etichetta, in modo tale che se si verifica una condizione che richiede di inviare il flusso all'etichetta corrispondente dopo la parola chiave Goto. Ecco un esempio:

```
Sub Test
   Dim num%
   num% = InputBox("Inserisci un numero inferiore a 5")

   If num% < 0 Then
      GoTo Nnum
   Else
      MsgBox (num%)
   End If

Nnum:
   MsgBox "Hai inserito un numero negativo"
End Sub
```

Allo stesso modo, è possibile creare un numero di etichette necessarie e fare riferimento ad esse quando si vuole. Il nome deve essere unico nel suo campo di applicazione e ciò significa che ogni etichetta deve avere un nome univoco nella stessa procedura. Ecco un esempio con varie etichette:

```
Sub Test()
   Dim risp As Byte

   risp = InputBox(" **__ Voci Multiple __** " & vbCrLf & vbCrLf & _
                   "Per creare una costante nel codice, " & _
                   "devi utilizzare la parola chiave Constant" & vbCrLf & _
                   "Cosa scegli (1=True/2=False)? ")
   If risp = 1 Then GoTo sb
   If risp = 2 Then GoTo gt

sb:
   MsgBox ("La parola chiave utilizzata per creare una costante è Const")
   GoTo Lv
gt:  MsgBox ("Constant non è una parola chiave")
Lv:
End Sub
```

Ecco un esempio di esecuzione del programma con risposta = 1:

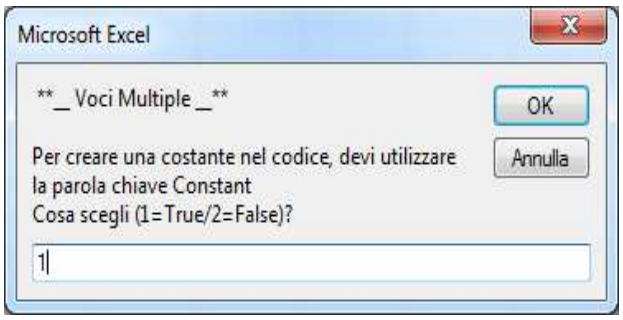

Fig. 10

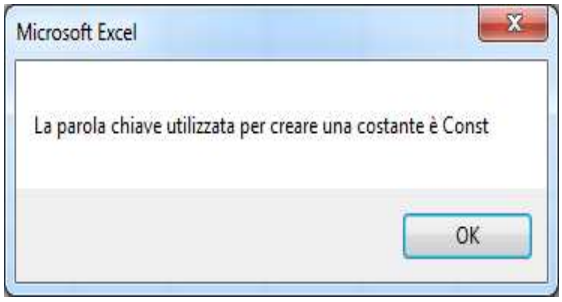

Fig. 11

Ecco un altro esempio di esecuzione dello stesso programma con risposta = 2:

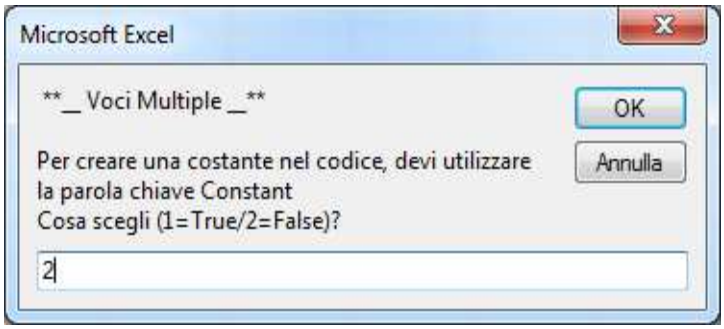

Fig. 12

Fig. 13

La negazione di una istruzione condizionale

Fino ad ora, abbiamo visto come un'istruzione condizionale che è vera o falsa, ma è possibile invertire il valore di una condizione, rendendolo Falso (o Vero). A sostegno di questa operazione, il linguaggio Visual Basic fornisce un operatore chiamato **Not**. La sua formula è:

Not Expression

Quando si scrive l'istruzione **Not**, deve essere seguita da un'espressione logica che può essere una semplice espressione booleana. Ecco un esempio:

```
Sub Test()
    Dim sposata As Boolean

    MsgBox ("La Signora è : " & sposata)
    MsgBox ("La Signora è : " & Not sposata)
End Sub
```

In questo caso, l'operatore Not viene utilizzato per modificare il valore logico della variabile e quando una variabile booleana è stata "negata", il suo valore logico è cambiato. Se il valore logico fosse stato Vero , sarebbe cambiato in False e viceversa, pertanto, è possibile invertire il valore logico di una variabile booleana di "Notting" o no "Notting" esso. Consideriamo ora il seguente programma:

```
Sub Test()
    Dim dsp As Boolean, tax As Double
    tax = 33#
    MsgBox ("Importo tassato : " & tax & "%")
    dsp = True
    If dsp = True Then
        tax = 30.65

        MsgBox ("Importo tassato : " & tax & "%")
    End If
End Sub
```

Questo produrrebbe:

Fig. 14

Fig. 15

Probabilmente il modo più classico di utilizzare l'operatore **Not** consiste in una "retromarcia" di un'espressione logica e per fare questo, si precedere l'espressione logica con l'operatore Not. Ecco un esempio:

```
Sub Test()
    Dim dsp As Boolean, tax As Double
    tax = 33#
    MsgBox ("Importo tassato : " & tax & "%")
    dsp = True

    If Not dsp Then
        tax = 30.65

        MsgBox ("Importo tassato : " & tax & "%")
    End If
End Sub
```

Questo produrrebbe:

Fig. 16

Allo stesso modo, è possibile negare qualsiasi espressione logica.