Introduzione alle Istruzioni condizionali

In alcuni compiti di programmazione, è necessario scoprire se una data situazione porta un valore valido e questa operazione viene fatta controllando una condizione. A supporto di questo, il linguaggio Visual Basic fornisce una serie di parole chiave e operatori che possono essere combinati per eseguire questo controllo, verificare se una condizione produce un risultato vero o falso.

Una volta che la condizione è stata verificata, è possibile utilizzare il risultato (True o False) per compiere delle azioni. Ci sono diversi modi per controllare una condizione, anche utilizzando diversi tipi di parole chiave per verificare cose diverse, essendo ben consapevoli di ciò che ciascuna parola chiave fa o non può fare per è importante scegliere quella giusta. La dichiarazione IF ... Then esamina la veridicità di un'espressione ed è strutturata in questo modo:

If CondizioneX Then Dichiarazione

Pertanto, il programma esamina una condizione, in questo caso *CondizioneX* che può essere una semplice espressione o una combinazione di espressioni e se CondizioneX è vera, allora il programma esegue la dichiarazione. Ci sono due modi per utilizzare la dichiarazione IF ... Then, se la formula condizionale è abbastanza breve, un modo è quello di scrivere su una riga, in questo modo:

```
Sub Test()
    Dim ricco As Boolean, tassa As Double
    tassa = 30
    MsgBox ("Imposta : " & tassa & "%")
    ricco = True

    If ricco = True Then tassa = 40.5
    MsgBox ("Imposta : " & tassa & "%")
End Sub
```

Questo listato produrrebbe:

Fig. 1

Fig. 2

Se per verificare la condizione ci sono molte istruzioni da eseguire, si dovrebbero scrivere le istruzioni su righe alternate, naturalmente, è possibile utilizzare questa tecnica anche se la condizione che si sta esaminando è breve. Se si scrive l'istruzione condizionale in più di una riga, è necessario terminare con *End IF* su una riga propria. La formula utilizzata è:

```
If CondizioneX Then
   Dichiarazione
End If
```

Ecco un esempio:

```
Sub Test()
   Dim ricco As Boolean, tassa As Double
   tassa = 30
   MsgBox ("Imposta : " & tassa & "%")
   ricco = True

   If ricco = True Then
   tassa = 40.5
   MsgBox ("Imposta : " & tassa & "%")
   End If
End Sub
```

E' inoltre possibile utilizzare il valore predefinito di una espressione booleana, infatti abbiamo visto che quando si dichiara una variabile booleana, per impostazione predefinita, viene inizializzata con il valore False. Ecco un esempio:

```
Sub Test()
   Dim ricco As Boolean
   MsgBox ("Sei ricco? " & ricco)
 End Sub
```

Questo produrrebbe:

Fig. 3

Sulla base di questo, se si desidera controllare se una variabile booleana dichiarata di recente e non inizializzata è uguale a False, è possibile omettere l'espressione di *= False*. Ecco un esempio:

```
Sub Test()
   Dim ricco As Boolean, tassa As Double
   tassa = 33
   If ricco Then tassa = 40.5
   MsgBox ("Percentuale tassabile : " & tassa & "%")
End Sub
```

Questo produrrebbe:

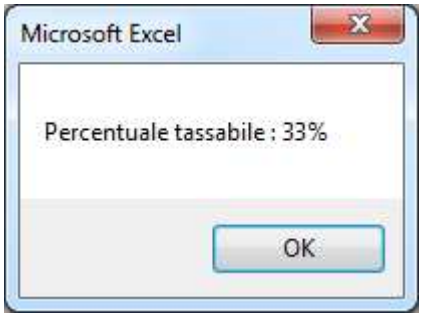

Fig. 4

Si noti che non ci sono simboli di uguale **(=)** dopo l'espressione *If ricco*, in questo caso, il valore della variabile è False, d'altra parte, se si vuole verificare se la variabile è True (Vera), assicuratevi di includere l'espressione = True. In generale, in caso di dubbio, è più sicuro inizializzare sempre la variabile e includere l'espressione = True o = False quando si valuta la variabile:

```
Sub Test()
   Dim ricco As Boolean, tassa As Double
   tassa = 33
   ricco = True
   If ricco = False Then tassa = 40.5
   MsgBox ("Percentuale tassabile : " & tassa & "%")
End Sub
```

Va ricordato che alcune funzioni booleane come *IsNumeric* e *IsDate*, il loro valore di default è True, questo significa che quando vengono richiamate è possibile omettere l'espressione = True. E' bene ricordare che quando una condizione è True o False la dichiarazione IF .. Then offre una sola alternativa: agire se la condizione è vera ed ogni volta che si desidera applicare un'espressione alternativa nel caso in cui la condizione è False, è possibile utilizzare la dichiarazione If .. Then .. Else. La formula di questa affermazione è:

```
If CondizioneX Then
   Dichiarazione1
Else
   Dichiarazione2
End If
```

Quando viene eseguita questa sezione di codice, se CondizioneX è True (Vera), allora viene eseguita l'istruzione contenuta in Dichiarazione1, se invece CondizioneX è False (Falsa), viene eseguita l'istruzione contenuta in Dichiarazione2. Ecco un esempio:

```
Sub Test()
   Dim eta As Integer, FasciaEta As String
   eta = 16
   If eta <= 18 Then
      FasciaEta = "Ragazzo"
   Else
      FasciaEta = "Adulto"
   End If
   MsgBox ("Categoria : " & FasciaEta)
End Sub
```

Ciò produrrebbe:

Fig. 5

IF Immediato

Per aiutarvi con il controllo di una condizione e la sua alternativa, il linguaggio Visual Basic fornisce una funzione chiamata **IIf** . La sua sintassi è:

```
Public Function IIf( _
   ByVal Expression As Boolean, _
   ByVal TruePart As Variant, _
   ByVal FalsePart As Variant _
) As Variant
```

Questa funzione opera come una condizione IF ... Then ... Else, prende tre argomenti richiesti e restituisce un risultato di tipo Variant e il valore restituito conterrà il risultato della funzione. Si deve tenere presente che la condizione da controllare viene passata come primo argomento alla funzione e se tale condizione è vera, la funzione restituisce il valore Vero e l'ultimo argomento viene ignorato, mentre se la condizione è falsa, il primo argomento viene ignorato e la funzione restituisce il valore del secondo argomento. Come già menzionato, è possibile recuperare il valore dell'argomento giusto e assegnarlo al risultato della funzione e il listato che abbiamo visto poco sopra può essere scritto come segue:

```
Sub Test()
    Dim eta As Integer, FasciaE As String
    eta = 16
    FasciaE = IIf(eta <= 18, "Ragazzo", "Adulto")
    MsgBox ("Categoria : " & FasciaE)
End Sub
```

Ciò produrrebbe lo stesso risultato che abbiamo visto in precedenza.

Scelta di un valore

Abbiamo imparato come controllare se una condizione è vera o falsa ed eseguire un'azione. Ecco un esempio:

```
Sub Test()
    Dim flag As Integer, tipo As String
    flag = 1
    tipo = "Sconosciuto"
    If flag = 1 Then
       tipo = "Tempo Pieno"
    End If
    MsgBox ("Tipo di impiego : " & tipo)
End Sub
```

Per fornire un'alternativa a questa operazione, il linguaggio Visual Basic fornisce una funzione chiamata **Choose**, la cui sintassi è:

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

Questa funzione richiede due argomenti richiesti, il primo è equivalente alla CondizioneX della formula If ... Then. Per la funzione Choose, questo primo argomento deve essere un numero ed è il valore rispetto al quale viene confrontato il secondo argomento. Prima di richiamare la funzione, è necessario conoscere il valore del primo argomento ed è possibile farlo dichiarando prima una variabile e inizializzarla con il valore desiderato. Ecco un esempio:

```
Sub Test()
    Dim flag As Byte, tipo As String
    flag = 1
    tipo = Choose(flag, ...)
    MsgBox ("Tipo di impiego : " & tipo)
End Sub
```

Il secondo argomento può essere la dichiarazione della nostra formula. Ecco un esempio:

Choose(flag, "Tempo Pieno")

Il secondo argomento è in realtà un elenco di valori e ogni valore ha una posizione specifica detta indice. Si tenga presente che per utilizzare la funzione in uno scenario If ... Then, si passa un solo valore come secondo argomento e questo valore (o argomento) ha un indice di 1. Quando la funzione Choose viene richiamata in un ciclo If, se il primo argomento contiene il valore 1, il secondo argomento viene convalidato. Quando la funzione Choose viene richiamata, restituisce un valore di tipo Variant ed è possibile recuperare quel valore, memorizzarlo in una variabile e usarlo all'occorrenza. Ecco un esempio:

```
Sub Test()
    Dim flag As Byte, tipo As String
    flag = 1
    tipo = Choose(flag, "Tempo Pieno")
    MsgBox ("Tipo di impiego : " & tipo)
End Sub
```

Ciò produrrebbe:

Fig. 6

In alcuni casi, la funzione Choose può produrre un risultato nullo, se consideriamo lo stesso listato usato in precedenza, ma con un diverso valore:

```
Sub Test()
    Dim flag As Byte, tipo As String
    flag = 2
    tipo = Choose(flag, "Tempo Pieno")
    MsgBox ("Tipo di impiego : " & tipo)
End Sub
```

Ciò produrrebbe un errore come in figura sotto riportata

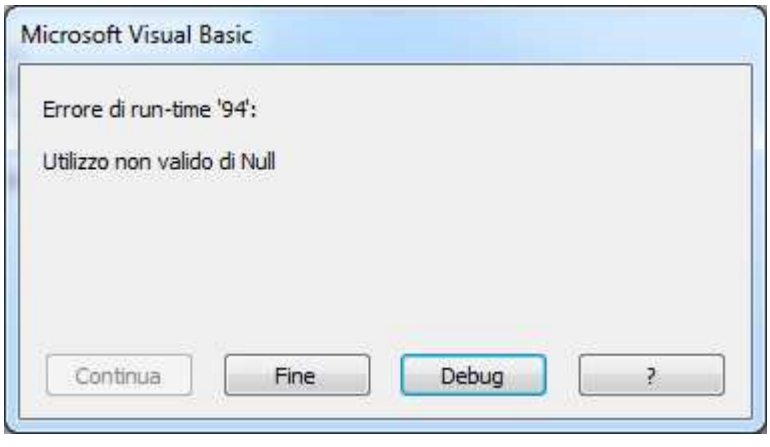

Fig. 7

In quanto non vi è alcun valore con indice 2 dopo la variabile che è stato inizializzato con il valore 2. Per utilizzare questa funzione come alternativa all'operazione If ... Then ... Else, si devono passare due valori per il secondo argomento in quanto ogni valore ha una posizione specifica in base al suo indice. Per utilizzare la funzione in una applicazione If ... Then ... Else, si devono passare due valori per il secondo argomento. Ecco un esempio:

Choose(Tipo di Impiego, "Tempo Pieno", "Part Time")

Il secondo argomento della funzione, che è il primo valore dell'argomentazione Choose, ha un indice di 1, mentre il terzo argomento della funzione, che è il secondo valore dell'argomento Choose, ha un indice di 2. In sostanza quando viene richiamata la funzione Choose, se il primo argomento ha il valore 1, il secondo argomento viene convalidato, mentre se il primo argomento ha il valore 2, viene convalidato il terzo argomento. Come già menzionato, è possibile recuperare il valore restituito della funzione, ecco un esempio:

```
Sub Test()
    Dim flag As Byte, tipo As String
    flag = 2
    tipo = Choose(flag, "Tempo Pieno")
    MsgBox ("Tipo di impiego : " & tipo)
End Sub
```

Ciò produrrebbe:

Fig. 8

Il passaggio di un valore

Come ulteriore alternativa a un If ... Then, il linguaggio Visual Basic fornisce una funzione denominata **Switch** . La sua sintassi è:

```
Public Function Switch( _
    ByVal ParamArray VarExpr() As Variant _
) As Variant
```

Questa funzione richiede un argomento obbligatorio e per utilizzarla in uno scenario If ... Then, si deve passare l'argomento come segue:

Switch(CondizioneX, Statement)

Dove CondizioneX è un segnaposto e si deve passare una espressione booleana che può essere valutata a Vero o Falso, se tale condizione è vera, il secondo argomento sarebbe stato ignorato. Quando la funzione Switch viene richiamata, si produce un valore di tipo Variant (come una stringa) che è possibile utilizzare all'occorrenza, ad esempio, è possibile memorizzarla in una variabile in questo modo:

```
Sub Test()
    Dim flag As Byte, tipo As String
    flag = 1
    tipo = "sconosciuto”
    tipo = Switch(flag = 1, "Part Time")
    MsgBox ("Tipo di impiego : " & tipo)
End Sub
```

In questo esempio, abbiamo utilizzato un numero come argomento, ma è anche possibile utilizzare un altro tipo di valore, come ad esempio una enumerazione. Ecco un esempio:

```
Private Enum tipo
    FullTime
    PartTime
    Stagionale
    Sconosciuto
End Enum

Sub Test()
    Dim flag As tipo, Result As String
    flag = tipo.FullTime
    Result = "Sconosciuto"
    Result = Switch(flag = tipo.FullTime, "Full Time")
    MsgBox ("Tipo di Impiego : " & Result)
End Sub
```

Quando si utilizza la funzione Switch, se si richiama con un valore che non è controllato dal primo argomento, la funzione genera un errore e per applicare questa funzione ad un ciclo If ... Then ... Else, è possibile richiamarla utilizzando la seguente formula:

Switch(CondizioneX1, Stato1, CondizioneX2, Stato2)

Dove il segnaposto CondizioneX1 passa una espressione booleana che può essere valutata a Vero o Falso, se tale condizione è vera, il secondo argomento sarebbe stato eseguito. Per fornire un'alternativa alla prima condizione, si può passare un'altra condizione come CondizioneX2 e se viene valutata come Vero, allora sarebbe stata eseguita l’istruzione2.