Corso di Visual Basic (Parte 3)

di Maurizio Crespi

Il corso dedicato alla programmazione con Microsoft Visual Basic continua con l'analisi della struttura di controllo Select Case e degli operatori logici fondamentali

Spesso si rivela necessario sviluppare dei programmi caratterizzati dalla capacità di eseguire istruzioni diverse in base al valore assunto da una o più variabili. A tal fine, Visual Basic prevede la struttura di selezione multipla *Select Case*, al centro dell'attenzione in questa terza parte del corso dedicato alla programmazione con il noto strumento Microsoft. Continua pertanto lo studio delle strutture di controllo, delle quali difficilmente un'applicazione può fare a meno.

Le soluzioni agli esercizi della scorsa puntata

Prima di affrontare lo studio di nuovi argomenti, è opportuno rivedere quanto esposto nello scorso numero. Come di consueto, lo spunto è offerto dalla correzione degli esercizi in esso proposti.

Primo esercizio

Il primo esercizio prevede la realizzazione di un programma in grado di leggere un numero mediante una textbox e di visualizzarne il valore assoluto. A tal fine, dopo aver creato un nuovo progetto, occorre aggiungere al form principale la casella di testo, a cui è possibile assegnare il nome *txtNumero,* nonché il pulsante *btnCalcola*, a cui è affidata la funzione di provocare l'aggiornamento del contenuto della label *lblRisultato* con il frutto dell'elaborazione. Ciò è ottenuto per mezzo della seguente procedura:

```
Private Sub btnCalcola_Click()
Dim StringaNumero As String
Dim Numero As Integer

StringaNumero = txtNumero.Text
Numero = Val(StringaNumero)
If Numero > 0 Then
      lblRisultato.Caption = "Valore assoluto: " & Numero
Else
      lblRisultato.Caption ="Valore assoluto:" & Str(-1 * Numero)
End If

End Sub
```

Il contenuto della textbox è posto nella variabile *StringaNumero*, che è convertita in un dato numerico per mezzo della funzione *Val*. Per fare in modo che il valore sia mantenuto invariato qualora risulti positivo, o ne sia calcolato l'opposto altrimenti, si impone il ricorso ad una struttura *If*, associata alla condizione

```
Numero > 0
```

Quando essa è verificata, il valore numerico fornito in ingresso è visualizzato senza subire modifiche. In caso contrario, la procedura provvede ad effettuare l'inversione di segno e ad assegnare il risultato, opportunamente trasformato in una stringa, alla label.

Secondo esercizio

Anche il secondo esercizio prevede la creazione di un programma in grado di ricevere un valore in ingresso per mezzo di una textbox e di visualizzare un risultato mediante una label. L'applicazione da sviluppare deve controllare che il dato inserito dall'utente rappresenti uno dei nomi dei giorni della settimana. Si desidera fare in modo che la verifica sia svolta automaticamente ogniqualvolta il contenuto della casella di testo subisca una variazione. Questa condizione è segnalata dall'oggetto, a cui si suppone di aver assegnato il nome *txtTestoDigitato,* con la generazione dell'evento *Change.* Per associare ad esso del codice, è sufficiente fare doppio clic con il mouse sulla textbox. In tal modo si crea la procedura *txtTestoDigitato_Change*, di seguito descritta.

```
Private Sub txtTestoDigitato_Change()
Dim Testo As String
Dim GiornoSettimana As Integer

Testo = UCase(txtTestoDigitato.Text)
If Testo = "LUNEDI" Then
GiornoSettimana = True
ElseIf Testo = "MARTEDI" Then
GiornoSettimana = True
ElseIf Testo = "MERCOLEDI" Then
GiornoSettimana = True
ElseIf Testo = "GIOVEDI" Then
GiornoSettimana = True
ElseIf Testo = "VENERDI" Then
GiornoSettimana = True
ElseIf Testo = "SABATO" Then
GiornoSettimana = True
ElseIf Testo = "DOMENICA" Then
GiornoSettimana = True
Else
GiornoSettimana = False
End If

If GiornoSettimana Then
lblRisultato.Caption = "E' un giorno della settimana"
Else
lblRisultato.Caption = "Non è un giorno della settimana"
End If

End Sub
```

Si noti l'uso della funzione *UCase*, che provvede a restituire la stringa derivante dalla conversione in lettere maiuscole del dato in ingresso. Ciò permette di verificare la corrispondenza del testo digitato dall'utente al nome di un giorno della settimana in modo indipendente dall'uso che è stato effettuato durante la sua scrittura delle lettere maiuscole. Il numero dei confronti, in questo caso, risulta elevato. Si ricorre pertanto all'uso della parola chiave *ElseIf*, al fine di semplificare la struttura di controllo, che in tal modo risulta organizzata in un unico livello. Essa provvede alla modifica della variabile intera *GiornoSettimana*, che può assumere solamente due valori. Il primo, corrispondente al valore logico *True*, è assegnato nel caso in cui il confronto con uno dei nomi dei giorni della settimana abbia fornito un esito positivo. In caso contrario, la variabile prende il valore *False*. Tale dato è oggetto di valutazione nella seconda struttura *If* che visualizza, facendo uso di una label a cui è stato assegnato il nome *lblRisultato*, un messaggio indicante l'esito della verifica. Si noti che la condizione è rappresentata dalla sola variabile *GiornoSettimana*, a cui non è associato alcun operatore di confronto. Ciò a prima vista può lasciare sconcertati. In realtà, è perfettamente lecito, in quanto è noto che la condizione che regola una struttura *If* deve essere di tipo booleano, ovvero deve poter assumere esclusivamente i valori *True* o *False*. Proprio come la variabile *GiornoSettimana*.

La struttura *Select Case*

L'esercizio precedente rappresenta un esempio di come una struttura *If* possa assumere dimensioni notevoli. In questi casi, la leggibilità del programma rischia di essere gravemente intaccata. Per evitare che questo accada, è possibile ricorrere ad una diversa struttura di controllo denominata *Select Case*, la cui sintassi è la seguente:

```
Select Case <variabile>
Case <valore 1>:
<blocco istruzioni 1>

[Case <valore 2>:
<blocco istruzioni 2>]
.
.
.
[Case <valore n>:
<blocco istruzioni n>]
[Case Else:
<istruzioni da eseguire se tutti i confronti falliscono>]
End Select
```

La struttura *Select Case* è adatta ad essere utilizzata ogniqualvolta si desideri variare il flusso del programma in base al risultato del confronto fra il valore di una variabile e uno o più dati costanti. Ognuno di essi deve essere preceduto dalla parola chiave *Case* e seguito dai due punti, nonché dal gruppo di istruzioni da eseguire quando il confronto ha esisto positivo. Inoltre, è possibile utilizzare la clausola *Case Else*, che va obbligatoriamente posta alla fine della struttura, per definire un gruppo di istruzioni che deve essere eseguito solo se

tutti i confronti hanno ottenuto un esito negativo. La procedura che costituisce la soluzione del secondo esercizio può quindi essere riscritta nel modo seguente:

```
Private Sub txtTestoDigitato_Change()
Dim Testo as string
Dim GiornoSettimana as integer
Testo = Ucase(txtTestoDigitato.text)

Select Case Testo
Case "LUNEDI":
GiornoSettimana = True
Case "MARTEDI":
GiornoSettimana = True
Case "MERCOLEDI":
GiornoSettimana = True
Case "GIOVEDI":
GiornoSettimana = True
Case "VENERDI":
GiornoSettimana = True
Case "SABATO":
GiornoSettimana = True
Case "DOMENICA":
GiornoSettimana = True
Case Else
GiornoSettimana = False

End Select

If GiornoSettimana Then
lblRisultato.caption = "E' un giorno della settimana"
Else
lblRisultato.caption = "Non è un giorno della settimana"
End If

End Sub
```

Il codice risulta in questo caso più leggibile. Tuttavia, c'è spazio per un ulteriore miglioramento, grazie alla possibilità di utilizzare la virgola per separare i valori a cui devono essere fatte corrispondere sequenze analoghe di istruzioni. La struttura *Select Case* può quindi essere nuovamente riscritta in modo estremamente più sintetico:

```
Select Case Testo
Case "LUNEDI", "MARTEDI", "MERCOLEDI", "GIOVEDI", "VENERDI",
"SABATO", "DOMENICA":
GiornoSettimana = True
Case Else
GiornoSettimana = False

End Select
```

È possibile pertanto notare come talvolta l'impiego della struttura *Case* possa agevolare notevolmente il compito del programmatore. Al fine di verificare ciò, si provi a realizzare un'applicazione che, ricevuto in ingresso un numero composto da una sola cifra, restituisca una stringa contenente l'indicazione dello stesso valore indicato in lettere.

Quando utilizzare la struttura Select Case

La struttura *Select Case*, potendo agire su una sola variabile per volta, presenta alcune limitazioni rispetto alla struttura *If*, in quanto non rende possibile l'esecuzione di valutazioni complesse. Si osservi il seguente esempio:

```
If (valore1 = 2) Then
Risultato = 5
ElseIf (valore2 = 3) Then
Risulato = 7
End if
```

Non è possibile eseguire le stesse operazioni mediante una sola struttura *Case*. Ne sono necessarie due nidificate, come illustrato di seguito:

```
Select Case Valore1
Case 2:
Risultato = 5
Case Else:
Select Case Valore2
Case 3:
Risultato = 7
End Select
End Select
```

La leggibilità del codice è in questo caso peggiorata. Inoltre, il numero delle righe digitate è aumentato. Da ciò appare evidente che l'uso della struttura *Case* non è sempre consigliabile. La sua opportunità deve essere valutata attentamente in fase di progettazione.

Confronti Multipli

Fino ad ora si è fatto uso della struttura di selezione multipla per confrontare il valore della variabile di controllo con alcuni dati costanti. In realtà, ci sono altre potenzialità nell'uso del costrutto *Case* che andiamo ora ad esplorare.

L'uso della struttura Case per valutare l'appartenenza a degli intervalli di valori.

Per mezzo della struttura *Case* diventa estremamente semplice valutare l'appartenenza del contenuto di una variabile a un intervallo di valori. Si supponga ad esempio di disporre di un form, caratterizzato anche questa volta dalla presenza di una label, denominata *lblRisultato*, di una casella di testo, a cui è assegnato il nome *txtValore* e di un pulsante avente la funzione di avviare

l'elaborazione. Si ipotizzi di voler avvisare mediante la label quando il valore indicato nella textbox, presunto numerico, risulta compreso fra 1 e 10, oppure se appartiene all'intervallo avente per estremi 11 e 100. Supponendo che il pulsante sia denominato *btn*Verifica, è possibile digitare la procedura:

```
Private Sub btnVerifica_Click()
Dim Testo As String
Dim Valore As Integer

If IsNumeric(txtValore.Text) Then
Valore = Val(txtValore.Text)
Select Case Valore
Case 1 To 10:
Testo = "E' compreso fra 1 e 10"
Case 11 To 100:
Testo = "E' compreso fra 11 e 100"
Case Else:
Testo = "Non appartiene agli intervalli"
End Select
Else
Testo = "Non è un valore numerico"
End If
lblRisultato.Caption = Testo

End Sub
```

Si noti l'uso della funzione *IsNumeric* atta a controllare che la stringa contenuta nella textbox, contenga effettivamente un numero (convertibile successivamente in un valore numerico per mezzo della funzione *Val*). Infatti, solo in questo caso sono effettuati i confronti. Altrimenti, si provvede a visualizzare nella label un opportuno messaggio di avviso. Questa volta, la struttura *Select Case* non fa riferimento a dei valori costanti, bensì a degli intervalli, ognuno dei quali è specificato per mezzo dei propri estremi, inframmezzati dalla parola chiave *To*.

L'uso della struttura Case per valutare se un valore è superiore ad una data soglia.
Si supponga di voler modificare l'esempio precedente per valutare se il valore inserito è superiore a 100. La struttura case deve essere modificata come segue:

```
Select Case Valore
Case 1 To 10:
Testo = "E' compreso fra 1 e 10"
Case 11 To 100:
Testo = "E' compreso fra 11 e 100"
Case Is > 100:
Testo = "E' maggiore di 100"
Case Else:
Testo = "Non appartiene agli intervalli"
End Select
```

Si noti l'introduzione della parola chiave *Is*, seguita da un'espressione di confronto. Se quest'ultima ha esito positivo, è eseguita l'istruzione associata.

Gli operatori logici elementari

Si ipotizzi di voler scrivere una struttura *If* per verificare se il valore di una variabile numerica intera, denominata *Numero*, appartiene all'intervallo compreso fra 10 e 100. È necessario valutare contemporaneamente due condizioni:

il numero deve essere maggiore di 10

il numero deve essere inferiore a 100

Per far sì che sia prodotta una stringa indicante l'esito del confronto, occorre digitare il seguente codice:

```
If Numero > 10 then
If Numero < 100 then
Testo = "Il numero è compreso fra 10 e 100"
Else
Testo = "Il numero non appartiene all'intervallo"
End if
End if
```

Si noti che l'uso della struttura *Case* avrebbe consentito la creazione di un listato più semplice e leggibile. È però possibile operare una semplificazione combinando i due confronti, ovvero creando un'unica espressione booleana in grado di stabilire l'appartenenza all'intervallo specificato. A tal fine, è necessario ricorrere agli operatori logici.

L'operatore And

Spesso si rivela necessario valutare la contemporanea validità di due o più condizioni.

È il caso dell'esempio precedente, in cui si desidera verificare se la variabile *Numero* contiene un valore maggiore di 10 e nel contempo minore di 100.

L'operatore *And* assolve questo compito, restituendo il valore logico *True* solo se le condizioni a cui è applicato sono contemporaneamente verificate. Il codice può pertanto essere riscritto come segue:

```
If (Numero > 10) And (Numero < 100) Then
Testo = "Il numero è compreso fra 10 e 100"
Else
Testo = "Il numero non appartiene all'intervallo"
End if
```

Come è possibile notare, la struttura risulta più semplice, in quanto composta da un solo livello.

L'operatore Or

A differenza del precedente, l'operatore *Or* restituisce il valore logico *True* se almeno una delle condizioni specificate è vera. Ad esempio, la condizione

```
(Numero = 5) Or (Numero > 11)
```

è verificata quando la variabile *Numero* assume un valore maggiore di 11 o uguale a 5.

L'operatore Not

Un altro utile operatore logico è quello di negazione (*Not*). Come è facile dedurre, esso restituisce il valore *True* se la condizione a cui è applicato non è verificata, mentre restituisce *False* in caso contrario. A titolo di esempio, si supponga di voler realizzare una struttura *If* in grado di generare una stringa indicate se il valore della variabile *Numero* risulta contemporaneamente diverso da 5 e da 10. Il codice da digitare è il seguente:

```
If Not ((Numero = 5) Or (Numero = 10)) Then
Testo = "Il numero è diverso da 5 e da 10"
End if
```

Si osservi la condizione. Essa risulta dalla combinazione di due confronti. Si tratta di

```
Numero = 5
```

e

```
Numero = 10
```

L'uso dell'operatore *OR* permette di verificare se almeno uno di essi ha esito positivo. In tal caso, non deve essere fornita alcuna indicazione. La stringa deve essere creata, infatti, solo quando entrambi i confronti hanno esito negativo, ovvero quando l'espressione

```
(Numero = 5) Or (Numero = 10)
```

restituisce il valore *False*, ovvero quando è verificata la condizione:

```
Not ((Numero = 5) Or (Numero = 10))
```

Come esercizio, si provi a realizzare un programma che riceva in ingresso una stringa contenente il nome di un mese e restituisca il numero dei giorni di cui esso si compone. Si preveda la possibilità di specificare l'anno a cui fare riferimento, al fine di offrire una corretta indicazione della lunghezza del mese di febbraio anche negli anni bisestili.

Conclusioni

Lo studio della struttura Select Case ha completato la panoramica sulle strutture di controllo iniziata nello scorso numero. Tali istruzioni ricoprono un ruolo fondamentale nella quasi totalità delle applicazioni. La loro conoscenza è inoltre indispensabile per la comprensione degli argomenti che verranno trattati nelle prossime puntate di questo corso. Al lettore va pertanto l'invito ad esercitarsi sui concetti illustrati e a non trascurare gli esercizi proposti.