Corso di Visual Basic (Parte 6) *di Maurizio Crespi*

Nella sesta parte, il corso dedicato alla programmazione in ambiente Microsoft Visual Basic punta il proprio obbiettivo su delle strutture dati di importanza fondamentale: i **_v e t t o r i_**

La fortuna dei calcolatori si deve alla capacità di effettuare in breve tempo delle operazioni ripetitive su una grande quantità di informazioni. Fino ad ora, in questo corso, i dati da elaborare sono sempre stati considerati contenuti all'interno di comuni variabili. Quando tuttavia il numero dei valori da gestire diventa elevato, esse si rivelano inadeguate a soddisfare le esigenze del programmatore. Lo scopo della lezione consiste nell'introdurre il concetto di vettore e di dimostrare quanto una struttura di questo tipo possa agevolare il compito dello sviluppatore in presenza di un'elevata mole di dati di tipo omogeneo. Prima di illustrare i nuovi concetti, è opportuno verificare la comprensione degli argomenti esposti nella precedente lezione. Lo spunto è come sempre rappresentato dalla discussione delle soluzioni degli esercizi in essa proposti.

Le soluzioni degli esercizi della lezione precedente

Nella scorsa lezione sono state descritte le strutture di iterazione basate sull'uso della parola chiave *Do*. Gli esercizi in essa proposti hanno perciò lo scopo di invitare il lettore a farne uso.

Primo esercizio

Il primo esercizio richiede la realizzazione di un programma che, ricevuta in ingresso una stringa alfabetica, indichi la posizione in essa occupata dalla prima lettera maiuscola, se presente. È evidente la necessità di implementare un ciclo che effettui una scansione della stringa, la cui lunghezza può variare indefinitamente e può anche essere nulla. Per questo motivo, occorre utilizzare una struttura che sia in grado di evitare le iterazioni qualora non siano necessarie. La scelta ricade sul ciclo *Do While*. Dopo aver realizzato un form dotato di una casella di testo, utilizzata per l'inserimento della stringa e di una label, mediante la quale è visualizzato il risultato dell'elaborazione, è possibile aggiungere un pulsante, avente lo scopo di avviare la ricerca. Esso deve pertanto essere in grado di rispondere all'evento *Click* con la seguente procedura:

```
Private Sub btnCalcola_Click()

Dim Stringa As String
Dim Carattere As String * 1
Dim i As Integer
Dim Lunghezza As Integer
Dim Fine As Boolean
```

```
Stringa = txtStringa.Text
Lunghezza = Len(Stringa)
i = 1
Fine = False

Do While Not (Fine) And (i <= Lunghezza)
    Carattere = Mid$(Stringa, i, 1)

    Select Case Carattere
       Case "A" To "Z":
         Fine = True
       Case Else:
       i = i + 1
    End Select

Loop

If Fine Then
  lblRisultato.Caption = "Posizione: " & i
Else
  lblRisultato.Caption = "Nessuna lettera maiuscola"
End If

End Sub
```

Il contenuto della casella di testo costituisce la stringa nella quale deve essere effettuata la ricerca. Per mezzo della funzione *Mid$* è estratto un carattere per volta. La sua posizione è definita dalla variabile *i*. Se il carattere risulta maiuscolo, la variabile booleana *Fine* è posta al valore logico *True* e la ricerca termina. In caso contrario, è incrementato il contatore di un'unità ed è eseguita una nuova iterazione per verificare il carattere seguente. Si noti che se il contatore assume un valore superiore al numero dei caratteri di cui è composta la stringa, condizione immediatamente verificata se la lunghezza è nulla, il ciclo termina. Si noti altresì l'uso della struttura *Select Case* per verificare se il carattere estratto risulta maiuscolo. Permettendo di definire degli intervalli, infatti, essa rende estremamente agevole l'operazione di controllo, rendendo inoltre il codice molto semplice da comprendere.

Secondo esercizio

Anche il secondo esercizio richiede la scansione di una stringa fornita dall'utente. La sua soluzione consiste infatti nella realizzazione di un programma in grado di ricevere in ingresso una stringa per mezzo di una textbox e di restituire la lunghezza della prima parola in essa contenuta. È possibile far uso di un form analogo a quello dell'esercizio precedente. Anche in questo caso, l'elaborazione è avviata dalla pressione del pulsante. Il codice che la descrive è quindi ancora contenuto nella procedura *btnCalcola_Click*:

```
Private Sub btnCalcola_Click()

Dim Stringa As String
Dim Carattere As String * 1
Dim i As Integer
Dim Lunghezza As Integer
Dim PosInizio As Integer
Dim Fine As Boolean

Stringa = txtStringa.Text
Lunghezza = Len(Stringa)
If Lunghezza > 0 Then
   i = 1
   Fine = False
   Rem Ricerca della posizione del primo carattere alfabetico
   Do
     Carattere = Mid$(Stringa, i, 1)
     Select Case Carattere
        Case "A" To "Z", "a" To "z":
        Fine = True
        Case Else:
        i = i + 1
     End Select
   Loop Until Fine Or (i > Lunghezza)

   If Fine Then
      Fine = False
      PosInizio = i
      Rem Ricerca di uno spazio o di un carattere di punteggiatura
      Rem che delimiti la parola
         Do While Not (Fine) And (i <= Lunghezza)
           Carattere = Mid$(Stringa, i, 1)
           Select Case Carattere
             Case " ", ",", ";", ".":
             Fine = True
             Case Else:
             i = i + 1
           End Select
         Loop
      lblRisultato.Caption = "Lunghezza prima parola:" & (i - PosInizio)
    Else
      lblRisultato.Caption = "La stringa non contiene parole"
    End If
Else
    lblRisultato.Caption = "Nessuna lettera maiuscola"
End If

End Sub
```

Dopo l'avvio, la procedura provvede a copiare il contenuto della textbox nella variabile denominata *Stringa*, di cui è calcolata la lunghezza per mezzo della funzione *Len*. Mediante un ciclo, è effettuata la ricerca del primo carattere alfabetico contenuto nella stringa. Se l'esito è positivo, ovvero se essa contiene almeno una parola, è utilizzato un ulteriore ciclo per ricercare il carattere che ne determina la fine. La differenza fra la posizione di quest'ultimo e quella del primo carattere costituisce la lunghezza della parola. Il valore è visualizzato per mezzo dell'etichetta testuale *lblRisultato*.

Perché servono i vettori

Si supponga di voler realizzare una semplice applicazione in grado di ordinare in modo crescente tre valori numerici. Non si tratta di un compito gravoso. È infatti sufficiente disegnare un form e porre su di esso tre textbox, necessarie per l'acquisizione dei valori, nonché una label destinata ad ospitare l'elenco ordinato. Per avviare l'elaborazione, è possibile ancora una volta utilizzare una procedura associata all'evento *Click* generato dalla pressione di un pulsante. Il codice è il seguente:

```
Private Sub btnOrdina_Click()

Dim Valore1 As Double
Dim Valore2 As Double
Dim Valore3 As Double
Dim temp As Double

Valore1 = Val(txtValore1.Text)
Valore2 = Val(txtValore2.Text)
Valore3 = Val(txtValore3.Text)

If Valore2 < Valore1 Then
   temp = Valore2
   Valore2 = Valore1
   Valore1 = temp
End If

If Valore3 < Valore1 Then
   temp = Valore3
   Valore3 = Valore1
   Valore1 = temp
End If

If Valore3 < Valore2 Then
   temp = Valore3
   Valore3 = Valore2
   Valore2 = temp
End If
lblRisultato.Caption = Valore1 & " - " & Valore2 & " - " & Valore3

End Sub
```

I dati forniti dall'utente, dopo essere stati convertiti in valori numerici, sono posti all'interno delle variabili *Valore1*, *Valore2*, *Valore3*. È successivamente effettuato un confronto fra i dati contenuti nelle variabili *Valore1 e Valore2*. Se il numero contenuto nella seconda è maggiore di quello memorizzato nella prima, i due valori sono scambiati. In modo analogo, è effettuato il confronto fra i numeri contenuti nelle variabili *Valore2* e *Valore3* ed è effettuato un ulteriore scambio qualora esso si riveli necessario per fare in modo che la variabile *Valore3* contenga un dato maggiore di quello posto in *Valore2*. Un ulteriore verifica è eseguita per fare in modo che il dato contenuto in *Valore3* sia maggiore di quello memorizzato in *Valore1*. In seguito all'esecuzione delle tre strutture *If*, si ottiene che:

il dato contenuto nella variabile *Valore2* risulta maggiore o uguale a quello posto in *Valore1*

il dato posto nella variabile *Valore3* risulta maggiore o uguale a quelli posti in *Valore2* e *Valore1*

La sequenza *Valore1*, *Valore2*, *Valore3*, risulta pertanto ordinata in modo crescente. I valori possono allora essere visualizzati utilizzando il carattere "-" come separatore. Sarebbe sicuramente molto meno agevole la realizzazione di un'applicazione analoga in grado di gestire 100 numeri. Ancor più ardua si rivelerebbe la creazione di un programma in grado di operare su 1000 o 10000 valori. È evidente che le comuni variabili non permettono di sviluppare in modo ragionevolmente semplice dei programmi in grado di gestire un numero elevato di informazioni dello stesso tipo. Per queste realizzazioni, è necessario far ricorso ai vettori.

I vettori

Fino ad ora, le variabili sono stare considerate come delle strutture in grado di associare a un nome simbolico un solo valore, secondo una relazione *uno a uno*. Saranno descritte ora delle speciali variabili, in grado di associare ad un unico nome uno o più valori, secondo una relazione *uno a molti*. Esse prendono il nome di vettori, o *Array*. Un array si presenta quindi come una variabile in grado di ospitare diversi valori in opportuni spazi numerati.

L'accesso ad un elemento contenuto in un vettore non può pertanto essere effettuato indicando il solo nome della struttura, bensì richiede che sia specificato anche il numero corrispondente alla posizione, che prende il nome di *indice*. Per meglio comprendere il concetto di vettore, si provi ad immaginare una variabile come una sorta di cassetto in cui è possibile riporre un'informazione.

Si supponga di applicare su questo cassetto un'etichetta su cui è riportata la scritta "Valori". Nel linguaggio naturale, per indicare a una persona di prendere il contenuto del cassetto caratterizzato dall'etichetta "Valori" e di porlo in un altro denominato "Risultato" si usa la frase: "Prendi il contenuto del cassetto *Valori* e

ponilo nel cassetto *Risultato*". In Visual Basic, ricordando che tali cassetti sono in realtà delle variabili, è possibile scrivere:

Risultato = Valori

Si supponga ora di disporre di un'intera cassettiera, in cui ogni singolo cassetto è numerato in modo univoco e di spostare su di essa l'etichetta. In questo caso, per accedere a un oggetto, occorre specificare il nome della cassettiera e il numero che identifica il cassetto che lo contiene. Ad esempio, è possibile dire "Prendi il contenuto del cassetto numero 3 della cassettiera *Valori* e ponilo nel cassetto di nome *Risultato*". Ciò in Visual Basic è tradotto con la riga:

Risultato = Valori(3)

Si noti che l'indice dell'elemento è posto fra parentesi tonde. Grazie ai vettori, è ora possibile realizzare in modo molto semplice un'applicazione in grado di richiedere 100 numeri e di visualizzare in un'etichetta di testo l'elenco dei valori ordinato in modo crescente. Il codice può essere associato all'evento *Load* del form di avvio, affinché sia eseguito durante la fase di creazione.

```
Private Sub Form_Load()

Dim Stringa As String
Dim Valori(100) As Double
Dim Temp As Double
Dim i As Integer
Dim j As Integer

Rem Acquisizione dei dati
For i = 1 To 100
  Stringa = InputBox("Inserire il numero")
  Valori(i) = Val(Stringa)
Next i

Rem Ordinamento con algoritmo di selezione diretta
For i = 1 To 99
For j = i + 1 To 100
If Valori(i) > Valori(j) Then
   Temp = Valori(i)
   Valori(i) = Valori(j)
   Valori(j) = Temp
End If

Next j, i
Rem Inserimento dei risultati nella textbox
lblRisultato.Caption = ""
For i = 1 To 100
  lblRisultato.Caption = lblRisultato.Caption & Valori(i) & Chr$(10)
Next i

End Sub
```

Si noti che la dichiarazione del vettore *Valori* è eseguita per mezzo dell'istruzione *Dim*, in modo pressoché analogo a quanto previsto per una qualsiasi variabile. L'unica differenza è costituita dalla presenza delle parentesi contenenti un numero che rappresenta la dimensione, ovvero il massimo indice utilizzabile. Il valore minimo per l'indice è sempre 0, tranne quando si provvede ad indicare nella sezione destinata delle dichiarazioni delle variabili globali la clausola

Option Base 1

In questo caso, l'indice minimo dei vettori è 1. Osservando la dichiarazione, si nota la presenza di un unico identificatore di tipo. Ciò rende immediatamente intuibile la necessità che tutti i dati presenti nel vettore siano omogenei. Nell'esempio, i valori sono acquisiti per mezzo della funzione *InputBox*, la cui sintassi è la seguente:

```
<stringa>=InputBox(<Messaggio di richiesta>
[, <titolo della finestra>]
[, <valore di default>]
[, <posizione orizzontale>]
[, <posizione verticale>])
```

Essa visualizza una finestra di dialogo che invita l'utente a digitare una stringa. Come è possibile notare, il programmatore deve specificare il messaggio di richiesta. Inoltre, egli può provvedere a un ulteriore personalizzazione indicando il titolo della finestra (in caso di omissione è utilizzato quello dell'applicazione), il valore di default, che è restituito se l'utente agisce sul pulsante *Annulla* e le coordinate orizzontale e verticale del punto di origine. Si noti che il riempimento del vettore avviene per mezzo di un ciclo *For*, che provvede a richiedere il valore di ogni singolo elemento richiamando 100 volte la funzione *InputBox*.

L'ordinamento per selezione diretta

Dopo l'acquisizione dei dati, è possibile provvedere al loro ordinamento. A tal fine, si fa uso dell'algoritmo detto di *selezione diretta*. Non si tratta del metodo più efficiente, ma si presta molto bene agli scopi didattici, in quanto è caratterizzato da un'estrema semplicità. Eccone una breve descrizione. Si supponga che *i* sia l'indice di un elemento del vettore. Affinché l'ordinamento sia crescente, è necessario che tutti gli oggetti caratterizzati dal possedere un indice superiore a *i* abbiano un valore maggiore di quello dell'elemento di indice *i*. Per fare in modo che ciò avvenga, per ogni posizione *j* maggiore di *i* è effettuato un confronto. Se l'elemento di indice *j* contiene un valore inferiore a quello dell'oggetto di posizione *i*, i dati sono scambiati. Al termine dell'iterazione, l'elemento di indice *i* contiene sicuramente un valore inferiore a tutti quelli che lo seguono. La ripetizione di questa operazione per tutti gli elementi del vettore ne provoca l'ordinamento. L'implementazione è basata sull'uso di due cicli *For* nidificati. Il più interno effettua la scansione del vettore alla ricerca dei valori inferiori a quello contenuto nell'elemento di indice *i*.

Il più esterno varia *i* per fare in modo che tale scansione sia ripetuta per tutti gli elementi del vettore, tranne l'ultimo, per il quale l'operazione sarebbe evidentemente inutile. La terza parte del programma è ancora una volta costituita da un ciclo, la cui funzione è quella di copiare nella label i dati contenuti nel vettore ordinato, separandoli con il carattere caratterizzato dal codice *ASCII* 10, che corrisponde al ritorno a capo. In tal modo i valori sono visualizzati su righe separate. Si noti l'uso della funzione *Chr$* per generare un carattere a partire dal suo codice *ASCII*. L'applicazione descritta si caratterizza per la presenza di ben quattro cicli. Ciò non è casuale. I vettori, infatti, essendo costituiti da dati di tipo omogeneo, sui quali in genere devono essere effettuate le stesse operazioni, si prestano molto bene ad essere utilizzati nelle procedure basate sull'uso delle strutture di iterazione.

Due esercizi

Per verificare la comprensione degli argomenti esposti, si provi ad eseguire i seguenti esercizi:

Primo esercizio

Si modifichi l'applicazione dell'esempio affinché effettui l'ordinamento in modo decrescente.

Secondo esercizio

Si realizzi un'applicazione in grado di estrarre a sorte 30 numeri, di calcolarne la media aritmetica e di visualizzare all'interno di una textbox solo quelli che risultano inferiori ad essa.

I vettori sono strutture di dati dall'importanza fondamentale, il cui uso si rivela spesso indispensabile per la realizzazione di programmi in grado di gestire grandi quantità di dati di tipo omogeneo. Nel prossimo numero si parlerà ancora degli array e saranno illustrati alcuni concetti più avanzati. Per la loro comprensione è quindi importante esercitarsi su quanto appreso questo mese.

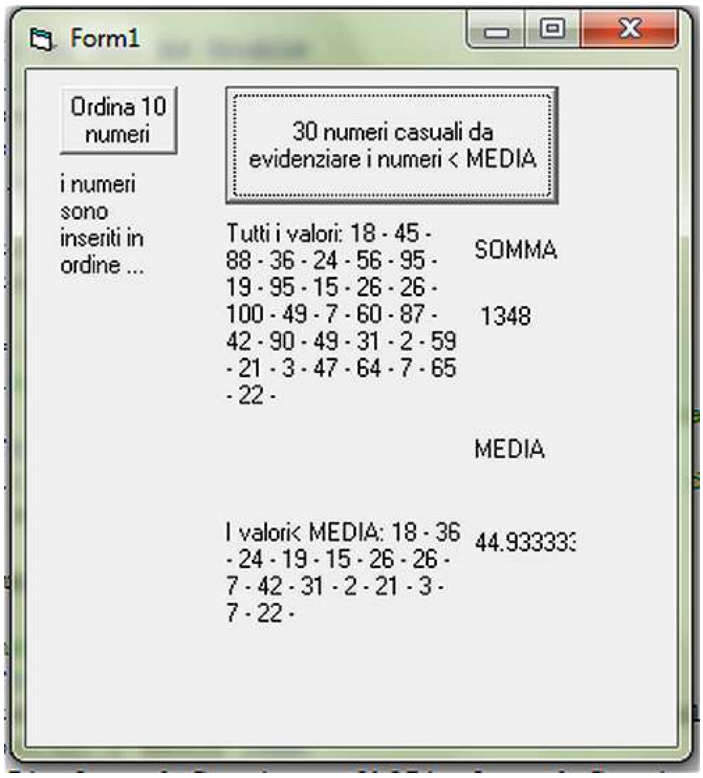
```
Private Sub btpCalcola_Click()

Dim Valori(1 To 30) As Double
Dim Temp, somma, media As Double
Dim i, j As Integer
Dim numero As Long

lblRisultato4.Caption = "Tutti i valori: "
lblRisultato1.Caption = "I valori< MEDIA: "
somma = 0
contatore = 0
Rem Acquisizione dei dati
Randomize 'ha lo scopo di inizializzare il generatore di numeri pseudo casuali

For i = 1 To 30
   Valori(i) = Int(100 * Rnd) + 1   'numeri casuali da 1 a 100
   somma = somma + Valori(i)
Next i
media = somma / 30

Rem Inserimento dei risultati nella textbox
For i = 1 To 30
  lblRisultato4.Caption = lblRisultato4.Caption & Valori(i) & " - "
    If Valori(i) < media Then
       lblRisultato1.Caption = lblRisultato1.Caption & Valori(i) & " - "
    End If
Next i
  lblRisultato3.Caption = Str$(somma)
  lblRisultato2.Caption = Str$(media)
End Sub
```



Rinfreschiamoci la

memoria...

Questa lezione si chiude in un modo un po' diverso, con una serie di domande aventi lo scopo di consentire un'autoverifica del livello di comprensione degli argomenti esposti in questi sei mesi. Le risposte corrette sono elencate di seguito, in corsivo.

Qual è la differenza che intercorre fra una textbox e una label?

La label permette solo di visualizzare una stringa. La textbox permette anche l'inserimento del dato da parte dell'utente.

A quale evento occorre associare del codice per fare in modo che sia eseguito in corrispondenza della pressione di un pulsante?

L'evento generato dalla pressione di un pulsante è denominato Click.

Può una variaibile di tipo Integer contenere il dato 23.45?

No, perché le variabili di tipo Integer non possono contenere numeri con cifre decimali.

Quale istruzione permette di dichiarare una variabile?

La dichiarazione di una variabile è possibile per mezzo dell'istruzione Dim.

È consentito utilizzare più strutture If nidificate?

Si, in Visual Basic non ci sono limiti al livello di nidificazione delle strutture If.

Quali istruzioni sono eseguite se la condizione che regola una struttura If è falsa?

Sono eseguite le istruzioni comprese fra le parole chiave Else e End If

A cosa serve la funzione Val?

La funzione Val converte una stringa in un valore numerico

Cosa restituisce l'operatore logico And applicato a due condizioni?

L'operatore logico And, se applicato a due condizioni, restituisce il valore True se e solo se entrambe sono verificate.

A cosa serve l'operatore logico Not?

L'operatore logico Not serve per negare una condizione.

A cosa serve la parola chiave Next?

La parola chiave Next è utilizzata per delimitare il blocco di istruzioni contenuto in un ciclo For.

A cosa serve la funzione Len?

La funzione Len restituisce la lunghezza della stringa fornitale come parametro

A cosa serve la parola chiave Step?

La parola chiave Step permette di specificare l'incremento della variabile di controllo ad ogni iterazione in un ciclo For. Se è omessa, l'incremento è di un'unità.

È possibile interrompere prematuramente un ciclo For?

Si, per mezzo dell'istruzione Exit For

Un ciclo While può ripetere un blocco di istruzioni per un qualsiasi numero di volte. Se la condizione non è verificata, può non eseguire alcuna iterazione.

Quando un ciclo Do Until ripete un blocco di istruzioni?

Un ciclo Do Until ripete un blocco di istruzioni fino a quando si verifica la condizione specificata dopo la parola Until

La clausola Option Base 1 serve per fare in modo che l'indice minimo dei vettori all'interno del progetto sia 1, anziché 0.