Gli Array

Se devo memorizzare più valori che sono in qualche modo parenti dal punto di vista logico, posso usare il concetto di ***vettore di variabili (array).***
Ad esempio, se devo memorizzare le temperature di tutti i giorni in un mese, non è particolarmente pratico definire 31 variabili:
Dim Temperatura1 As Single
Dim Temperatura2 As Single
...
Definisco allora un vettore di Single, fatto da 31 elementi:

Dim Temperatura(30) As Single

Il numero di elementi viene cioè indicato tra parentesi tonde, e prende il nome di Subscript. Ho scritto 30 perché gli elementi sono per impostazione definita numerati a partire da 0 (e quindi il trentunesimo elemento ha il numero 30).
Ai singoli elementi si accede semplicemente usando il loro ***indice***:
Temperatura(0) = 23.7
Temperatura(Giorno) = 25 'L'indice può essere una variabile (in questo caso Giorno), o anche un'espressione o come nel primo esempio, semplicemente possiamo indicare il numero corrispondente alla posizione.
Se poi le temperature da memorizzare sono la minima e la massima, allora di vettori me ne servono due, o meglio mi serve una matrice di 31 righe per 2 colonne (in inglese è ancora array):
Dim Temperatura(30,1) As Single
Se non so a priori quanti elementi conterrà il mio array, posso dichiararlo senza precisarlo:
Dim A() As Single
Prima di usarlo, devo dichiarare il numero di elementi, mediante l'enunciato ReDim:
ReDim A(30)
Il bello è che ora posso ridimensionare tutte le volte che voglio il mio array, purché cambi solo il numero di elementi, e non il numero di dimensioni che ho dato al primo uso di ReDim . Basta che scriva:
ReDim A(60)
(ma non posso scrivere ReDim A(60,2) perché ***non posso aggiungere a posteriori un'altra dimensione.***)

Usando ReDim senza nient'altro, perdo tutti valori che avessi già memorizzato nell'array (essa ritorna piena di 0 se fatta di tipi numerici, o di stringhe nulle se di tipo String).
Se, come spesso accade quando si ingrandisce un'array, voglio mantenere i valori precedenti, devo usare la parola chiave aggiuntiva ***Preserve***:
ReDim Preserve A(60)
Sia ReDim sia Preserve possono essere usati anche per diminuire il numero di

elementi (in tal caso naturalmente, anche se uso Preserve, perdo gli elementi che decido di togliere).

Vediamo ora un esempio di progetto in cui si prevede la memorizzazione delle temperature giornaliere di una settimana in un array e di produrre in output la temperatura media.

```
Dim temperature(6), cont, tottemperature As Integer
Dim media As Single

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    For cont = 0 To 6
        temperature(cont) = InputBox(" inserire la temperatura del " & cont + 1 & "° giorno")
    Next cont
    For cont = 0 To 6
        tottemperature = tottemperature + temperature(cont)
    Next cont
    media = tottemperature / 7
    MsgBox("la media delle temperature è" & media)
End Sub
```

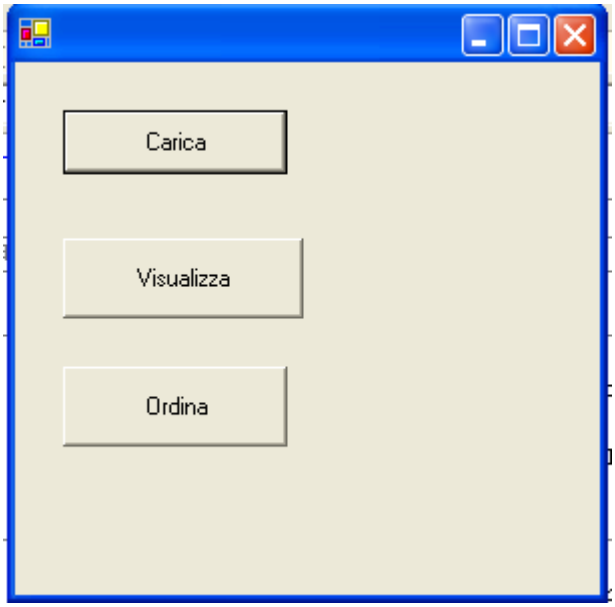
Quando si cominciano ad utilizzare gli array, nasce l'esigenza di ordinare gli elementi che lo compongono,***(SORT)*** o di ricercare un elemento all'interno dell'array stesso ***(RICERCA)*** Esistono vari algoritmi sia per l'ordinamento che per la ricerca.

ORDINAMENTO CON METODO DI SOSTITUZIONE

Questo metodo è forse uno dei più semplici, ma visto che prevede lo scorrimento di tutto l'array più volte, è adatto nei casi in cui l'array è di dimensioni contenute. In pratica, partendo dal primo elemento, ogni elemento viene confrontato con tutti gli altri e laddove risultasse maggiore viene scambiato di posizione con il successivo (questo in caso di ordinamento crescente). La stessa procedura viene utilizzata indifferentemente per ordinare un array di valori numerici o di stringhe. Per effettuare lo scambio di posizione, si utilizza una variabile di comodo che nell'esempio abbiamo chiamato ***temp.***

```
Dim V(4), i, j, temp As Integer
Dim stringa As String

Private Sub btnCarica_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCarica.Click
    For i = 1 To 10
        V(i) = InputBox(" inserisci il " & i & "° valore")
    Next i
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    stringa = ""
    For i = 1 To 10
        stringa = stringa + CStr(V(i)) + ";"
    Next i
    MsgBox(stringa)
End Sub

Private Sub btnOrdina_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOrdina.Click
    For i = 1 To 9
        For j = i + 1 To 10
            If V(i) > V(j) Then
                temp = V(i)
                V(i) = V(j)
                V(j) = temp
            End If
        Next j
    Next i
End Sub
```

In questo esempio vedete come con tre semplici procedure è possibile prevedere il caricamento di un array, la visualizzazione e l'ordinamento di un array di numeri. Ovviamente una volta ordinato basterà cliccare ancora su Visualizza per vedere l'array ordinato.

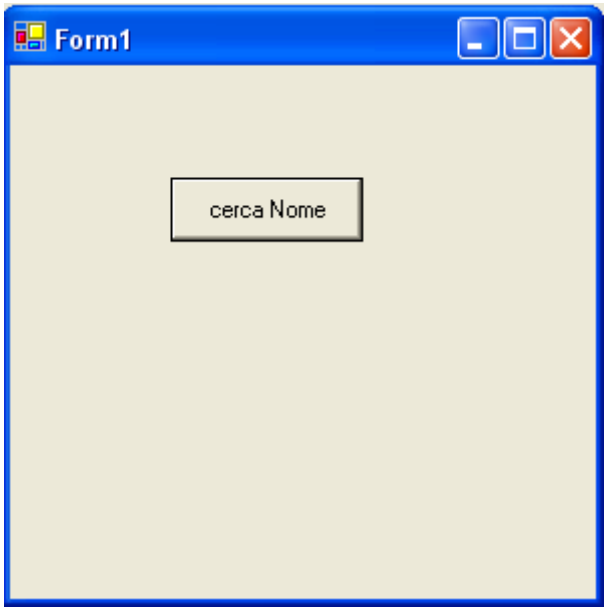


RICERCA SEQUENZIALE

La ricerca sequenziale consiste nel leggere tutte le componenti degli array, confrontarle con il valore dell'elemento cercato e nel caso in cui si trovi

corrispondenza tra i due elementi, si interromperà la ricerca, altrimenti, si procederà fino alla fine. Nell' esempio che vi propongo, dopo aver caricato i nomi il sesso e l'età di n elementi in due array paralleli, ho previsto la ricerca per nome, restituendo TRUE (vero) se l'elemento inserito è presente nell'array, con le relative informazioni contenute negli array paralleli, età e sesso.

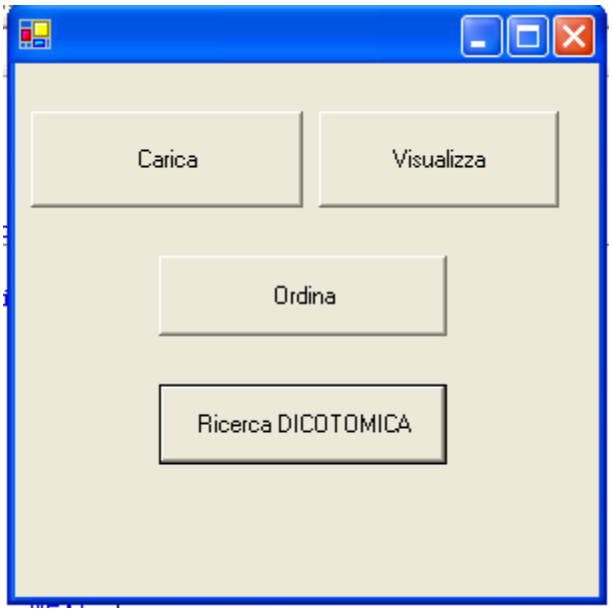
```
Codice generato da Progettazione Windows Form
    Dim n(3), s(3), nc, sc As String
    Dim età(3), i As Integer
    Dim trovato As Boolean = False
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        For i = 1 To 3
            n(i) = InputBox("inserire il nome")
            s(i) = InputBox("inserire il sesso m/f")
            età(i) = InputBox(" inserire l'età")
        Next i
    End Sub

    Private Sub Btnn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnn.Click
        nc = InputBox(" inserire il nome da cercare")
        For i = 1 To 3
            If nc = n(i) Then

                trovato = True
                MsgBox("il nome è " & trovato & " ed è  " & nc & " il suo sesso è  " & s(i) & " la sua età è "
            End If
        Next i
    End Sub
```



RICERCA DICOTOMICA

E' un altro metodo di ricerca che può essere utilizzato solo se l'array è ordinato. Si procede dividendo l'array in due parti ed escludendo dal controllo la parte dell'array che sicuramente non conterrà l'elemento cercato. La divisione in parti dell'array, procede fino a quando non viene trovato l'elemento cercato oppure fino a che rimarranno 2 soli elementi e sarà quindi opportuno inviare il messaggio "ELEMENTO NON PRESENTE IN ARRAY" Qui di seguito vi mostro il codice per

integrare l'esercizio già visto precedente con un bottone per la ricerca dicotomica.




```
Private Sub BtnDICOTOMICA_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnDICOTOM
    cercato = InputBox("inserisci l'elemento da cercare...")

    trovato = False
    sx = 0
    dx = 10 - 1

    Do
        md = (sx + dx) \ 2
        If V(sx) = cercato Or V(dx) = cercato Or V(md) = cercato Then
            trovato = True
        Else
            If V(md) < cercato Then
                sx = md + 1
            Else
                dx = md - 1

            End If

        End If
    Loop Until sx > dx Or trovato = True
    If trovato = True Then
        MsgBox("ELEMENTO PRESENTE NELL'ARRAY")
    Else
        MsgBox("ELEMENTO non PRESENTE NELL'ARRAY")
    End If

End Sub
```

LE MATRICI

Per ***Matrice*** si intende un array a due (o più dimensioni), cioè un insieme di elementi dello stesso tipo individuabili all'interno della matrice da una coppia di indici, uno per la riga e uno per la colonna. Quindi avremo in fase di dichiarazione

dim m(nr,nc) as integer per dichiarare una matrice dove nc indica il valore massimo che può avere l'indice j (colonne) e nr il valore massimo di i (righe).

Il caricamento di una matrice prevederà quindi l'uso di due cicli for :

```
for i = 0 to nr
   for j = 0 to nc
      m(i,j)=inputbox("inserisci elemento")
   next j
next i
```