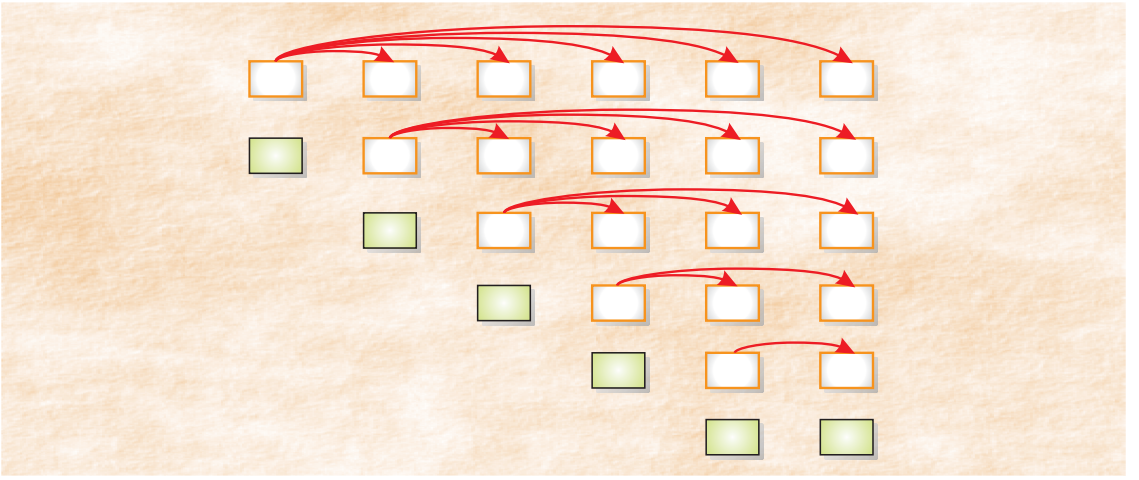
Algoritmo per l'ordinamento di un array

Spesso si ha la necessità di lavorare con **vettori ordinati**, cioè si devono mettere gli elementi in ordine alfabetico se sono stringhe o in in ordine di grandezza se sono valori numerici. Ordinare gli elementi (in inglese **sort**) non è un'operazione semplice, soprattutto nel caso in cui la dimensione del vettore sia grande.
In Visual Basic l'ordinamento di un array si può realizzare con il metodo **Sort** della classe **Array**:

Array.Sort (*vettore*)

Esistono molti algoritmi di *sort*. Qui viene presentato uno dei più semplici: il **metodo di sostituzione** che è efficace per un numero piccolo di elementi da ordinare. Poiché il tipo di dati su cui si opera è ininfluente, viene descritto il procedimento operando su un array di numeri interi.
Questo metodo consiste nel confronto di ciascun elemento, a cominciare dal primo fino all'ultimo, con tutti gli altri elementi che sono disposti nelle posizioni successive a quella in esame. Quando tra gli elementi successivi all'elemento in esame si incontra un elemento più piccolo, si procede allo scambio tra i due elementi, in modo che, alla fine del confronto di ciascun elemento con i successivi, si ottenga nella posizione in esame il valore minimo rispetto ai successivi. Alla fine gli elementi, dal più piccolo al più grande, compaiono ordinati all'interno dell'array a partire dal primo.



Per capire meglio l'algoritmo si veda il seguente esempio.
Si deve ordinare il vettore:

Valore	3	2	6	4	5
Indice	0	1	2	3	4

L'algoritmo inizia prendendo in considerazione il primo elemento (3) e confrontandolo con tutti i successivi. Subito trova che 2 < 3 per cui si procede allo scambio:

Valore	2	3	6	4	5
Indice	0	1	2	3	4

Si continua il confronto tra il primo elemento (diventato ora 2) e gli altri (6, 4 e 5), ma non si trovano elementi minori di 2.

L'algoritmo passa ora a confrontare il secondo (il numero 3) con tutti i successivi (6, 4, e 5) che sono tutti maggiori di 3, per cui non si opera alcuno scambio.

Si passa al terzo (il 6) e si vede che il quarto (4) è minore, per cui si effettua lo scambio:

Valore	2	3	4	6	5
Indice	0	1	2	3	4

Si continua il confronto tra l'elemento di posto 2 (che ora è diventato il 4) e il restante (il 5) ma non si trovano elementi minori.

Si passa al quarto (6) che è maggiore del quinto (5) per cui si effettua lo scambio:

Valore	2	3	4	5	6
Indice	0	1	2	3	4

Tutti i confronti sono stati effettuati, per cui l'algoritmo termina. Il vettore risulta ordinato.

Riassumendo l'esempio, i confronti da fare sono stati (vengono indicati gli indici):

- (0 con 1), (0 con 2), (0 con 3), (0 con 4)
- (1 con 2), (1 con 3), (1 con 4)
- (2 con 3), (2 con 4)
- (3 con 4)

Quindi 4+3+2+1=10 scambi. Nel caso di un vettore con n elementi gli scambi saranno

$$(n\text{-}1)+(n\text{-}2)+(n\text{-}3)+\ldots+3+2+1=n*(n\text{-}1)/2$$

I confronti da fare risultano numerosi: per un vettore di 100 componenti ne occorrono quasi 5000. L'algoritmo è molto semplice, ma è utilizzabile solamente per vettori di piccole dimensioni.

Progetto

Ordinare un array di numeri con il metodo di sostituzione.

Si deve creare il programma che permette di caricare e di ordinare le componenti di un vettore utilizzando il metodo di sostituzione. Vengono riutilizzati gli algoritmi *Carica* e *Mostra* del progetto precedente.

Dati di input:
dimensione del vettore
componenti del vettore

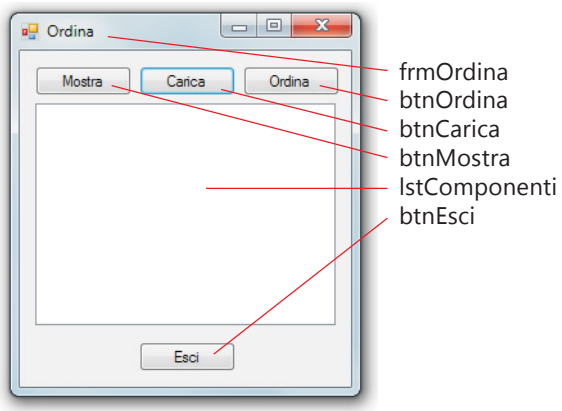
Dati di output:
vettore ordinato.

Nome del progetto
Sostituzione di tipo *Applicazione Windows Form*

Disegno dell'interfaccia grafica

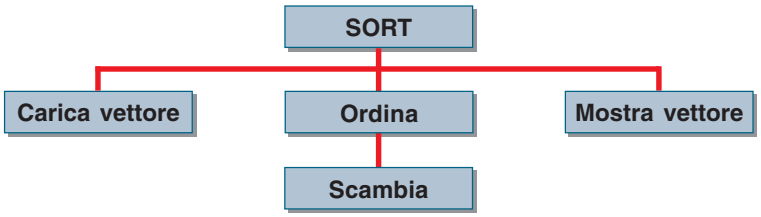
Sono presenti i controlli del progetto precedente che permettono di caricare e di visualizzare il vettore. In aggiunta è presente il pulsante di comando *Ordina* per eseguire l'ordinamento del vettore.

Classe	Proprietà dell'oggetto	
Form	Name	frmOrdina
	Text	*Ordina*
ListBox	Name	lstComponenti
Button	Name	btnOrdina
	Text	*Ordina*
Button	Name	btnMostra
	Text	*Mostra*
Button	Name	btnCarica
	Text	*Carica*
Button	Name	btnEsci
	Text	*Esci*



Gestione degli eventi

Per lo sviluppo completo del programma occorre distinguere tre sottoproblemi e un ulteriore sottoproblema per lo scambio, secondo lo schema gerarchico:

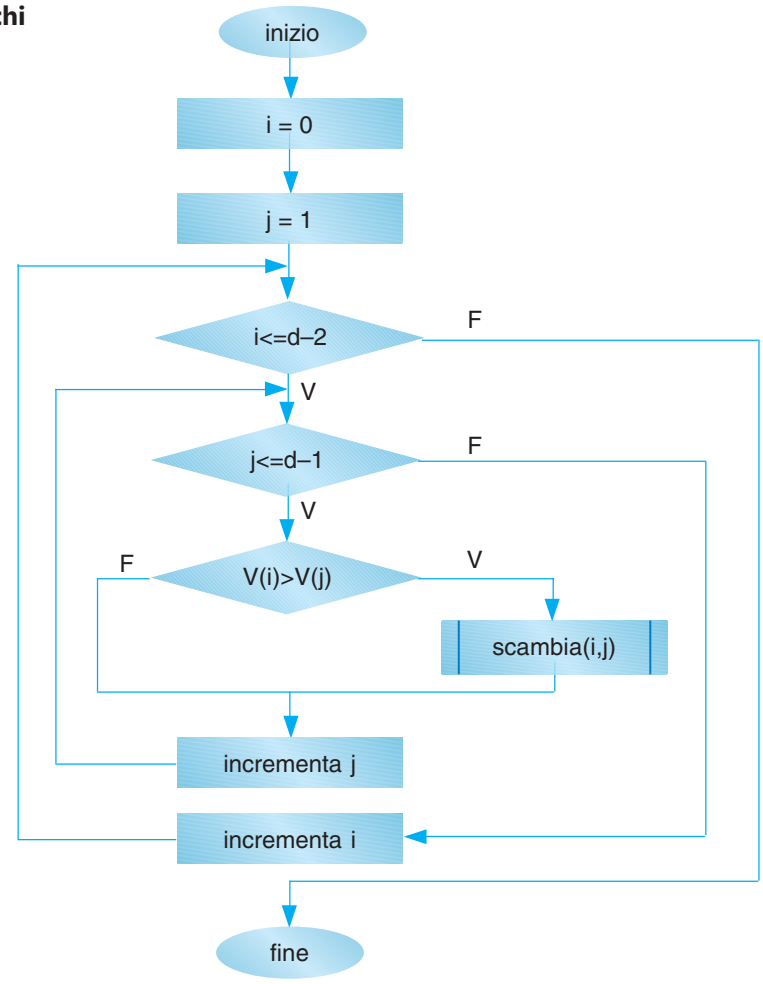


Occorre organizzare due iterazioni enumerative nidificate l'una rispetto all'altra. Nell'iterazione più esterna viene percorso tutto il vettore fino al penultimo elemento, mentre la più interna prende in considerazione tutti gli elementi a destra di quello individuato dall'iterazione esterna. Se il primo elemento risulta minore del secondo si procede allo scambio. Il problema in esame contiene, quindi, al suo interno il sottoproblema dello scambio del contenuto di due variabili.

Algoritmo in pseudocodifica

```
inizio
    per i da 0 a dimensione - 2
        per j da i+1 a dimensione - 1
            se vettore(i) > vettore(j)
            allora
                scambia(i,j)
            fine se
        ripeti
    ripeti
fine
```

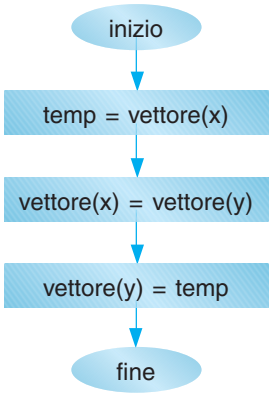
Diagramma a blocchi



Algoritmo in pseudocodifica scambia(x, y)

```
inizio
    assegna temp = vettore(x)
    assegna vettore(x) = vettore(y)
    assegna vettore(y) = temp
fine
```

Diagramma a blocchi

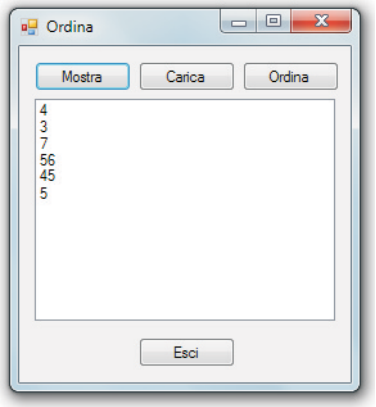


Codice Visual Basic

```
   Const Max As Integer = 10
   Dim dimensione As Integer = 0
   Dim vettore() As Integer

   Private Sub btnCarica_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCarica.Click
      Do
         dimensione = InputBox("Inserisci la dimensione", "Chiedi dimensione")
      Loop Until dimensione >= 1 And dimensione <= Max
      ReDim vettore(dimensione - 1)
      For i = 0 To dimensione - 1
         vettore(i) = InputBox("Inserisci il " & i + 1 & "° elemento",
"Inserisci")
      Next
   End Sub

   Private Sub btnMostra_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMostra.Click
      lstComponenti.Items.Clear()
      For Each valore In vettore
         lstComponenti.Items.Add(valore)
      Next
   End Sub

   Private Sub scambia(ByVal x As Integer, ByVal y As Integer)
      Dim temp As Integer
      temp = vettore(x)
      vettore(x) = vettore(y)
      vettore(y) = temp
   End Sub

   Private Sub btnOrdina_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnOrdina.Click
      For i = 0 To dimensione - 2
         For j = i + 1 To dimensione - 1
            If vettore(i) > vettore(j) Then
               scambia(i, j)
            End If
         Next
      Next
   End Sub

   Private Sub btnEsci_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEsci.Click
      End
   End Sub
End Class
```

Il codice associato ai pulsanti di comando *Carica*, *Mostra* e *Fine* è identico a quello del Progetto 1.

Prova di esecuzione
Si inseriscono i dati come nel Progetto 1. Facendo clic sul pulsante *Mostra*, si visualizzano i dati per verificare che non siano già ordinati.



Facendo clic sul pulsante *Ordina* i dati vengono ordinati.
Per verificare il corretto ordinamento è necessario fare clic di nuovo sul pulsante *Mostra*.