7.11 Gestione di una sequenza

In questo paragrafo consideriamo il problema di far gestire all'utente una o più sequenze di interi mediante il seguente menu:

```
GESTIONE SEQUENZA


1. Immissione
2. Ordinamento
3. Ricerca completa
4. Ricerca binaria
5. Visualizzazione
0. fine

Scegliere una opzione:
```

Le opzioni possono essere scelte un numero di volte qualsiasi, finché non si seleziona la numero zero, che fa terminare il programma. Ovviamente, prima di tutto si deve scegliere la prima opzione per immettere la sequenza, ma successivamente questa possibilità può essere sfruttata per lavorare su altre sequenze.
Nel Listato 7.9 proponiamo il programma completo; dato che tutti gli algoritmi relativi sono stati visti nel Capitolo 5, adesso ci soffermiamo soltanto sull'uso delle funzioni e sul passaggio dei parametri.
L'array che conterrà la sequenza viene dichiarato come variabile globale:

```
int vet[MAX_ELE];  /* array che ospita la sequenza */
```

dunque tutte le funzioni del file vi possono accedere. (Nel Capitolo 9 vedremo una soluzione migliore.)
Decidiamo di far svolgere il compito di visualizzare il menu e gestire le scelte dell'utente alla funzione `gestione_sequenza`; essa dunque non dovrà restituire nessun valore e non accetterà nessun parametro:

```
void gestione_sequenza( void );
```

ma verrà semplicemente invocata dal `main`:

```
gestione_sequenza();
```

Viene naturale, poi, far corrispondere a ogni opzione una funzione che svolga il compito stabilito. Nel caso sia selezionata l'opzione 1 essa viene immessa nella variabile intera `scelta` e per mezzo del costrutto `switch-case` è mandata in esecuzione la funzione `immissione`:

```
case 1: n = immissione();
```

Essa deve ritornare a `gestione_sequenza` il numero di valori immessi in modo che esso possa essere reso noto alle altre funzioni; tale valore viene memorizzato nella variabile `n`. Se si verifica la dichiarazione di `immissione` si può vedere che effettivamente essa ritorna un intero.
Alla funzione `ordinamento` deve essere passato il numero di elementi della sequenza:

```
case 2: ordinamento( n );
```

Anch'essa agisce sulla variabile generale `vet[MAX_ELE]` ordinando i suoi elementi e non restituisce alcun valore: infatti il suo valore di ritorno è descritto in fase di dichiarazione e di definizione come `void`.

Nel caso di scelta 3 alla funzione `ricerca` deve essere passato, oltre alla lunghezza della sequenza, anche il valore dell'elemento da ricercare precedentemente richiesto all'utente in `gestione_sequenza`:

```
posizione = ricerca( n, ele );
```

La funzione ritorna un valore intero, che corrisponde alla posizione dove è stato reperito l'elemento. Considerazioni analoghe valgono per la funzione di ricerca binaria `ric_bin`.

```
#include <stdio.h>

#define MAX_ELE 1000      /* massimo numero di elementi */
int vet[MAX_ELE];         /* array che ospita la sequenza */

void gestione_sequenza( void );
int immissione( void );
void ordinamento( int );
int ricerca( int, int );
int ric_bin( int, int );
void visualizzazione( int );

main()
{
gestione_sequenza();
}

void gestione_sequenza()
{
int n;
int scelta = -1;
char invio;
int ele, posizione;

while(scelta != 0) {
 printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
 printf("\t\t\t GESTIONE SEQUENZA");
 printf("\n\n\n\t\t\t 1. Immissione");
 printf("\n\n\t\t\t 2. Ordinamento");
 printf("\n\n\t\t\t 3. Ricerca completa");
 printf("\n\n\t\t\t 4. Ricerca binaria");
 printf("\n\n\t\t\t 5. Visualizzazione");
 printf("\n\n\t\t\t 0. fine");
 printf("\n\n\n\t\t\t\t Scegliere una opzione: ");
 scanf("%d", &scelta);
 scanf("%c", &invio);
 printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");

 switch(scelta) {
   case 1: n = immissione();
```

```
            break;
   case 2: ordinamento( n );
            break;
   case 3: printf("Elemento da ricercare: ");
            scanf("%d", &ele);
            scanf("%c", &invio);
            posizione = ricerca( n, ele );
            if(ele == vet[posizione])
              printf("\nElemento %d presente in posizione
                      %d\n", ele, posizione);
            else
              printf("\nElemento non presente!\n");
            printf("\n\n Premere Invio per continuare...");
            scanf("%c", &invio);
            break;
   case 4: printf("Elemento da ricercare: ");
            scanf("%d", &ele);
            scanf("%c", &invio);
            posizione = ric_bin( n, ele );
            if(posizione != -1)
              printf("\nElemento %d presente in posizione
                      %d\n", ele, posizione);
              else
              printf("\nElemento non presente!\n");
            printf("\n\n Premere Invio per continuare...");
            scanf("%c", &invio);
            break;
   case 5: visualizzazione( n );
            break;
 }
}
}

int immissione()
{
int i, n;
char invio;
do {
  printf("\nNumero elementi: ");
  scanf("%d", &n);
}
while (n < 1 || n > MAX_ELE);

for(i = 0; i < n; i++) {
  printf("\nImmettere un intero n.%d: ",i);
  scanf("%d", &vet[i]);
}
return( n );
}

void ordinamento( int n )
{
int i, p, k, n1;
int aux;
p = n; n1 = p;
do {
    k = 0;
    for(i = 0; i < n1-1; i++)
      if(vet[i] > vet[i+1]) {
```

```
        aux = vet[i]; vet[i] = vet[i+1];
        vet[i+1] = aux;
        k = 1; p = i + 1;
      }
    n1 = p;
}
while (k == 1);
}

/* Ricerca sequenziale */
int ricerca ( int n, int ele )
{
int i;
i = 0;
while (ele != vet[i] && i < n-1) ++i;
return( i );
}

/* ricerca binaria */
int ric_bin( int n, int ele )
{
int i, alto, basso, pos;
alto = 0; basso = n - 1; pos = -1;
do {
  i = (alto+basso)/2;
  if(vet[i] == ele) pos = i;
  else if(vet[i] < ele) alto = i + 1;
         else basso = i - 1;
}
while(alto <= basso && pos == -1);
return( pos );
}

void visualizzazione( int n )
{
int i;
char invio;
for(i = 0; i < n; i++)
  printf("\n%d", vet[i]);
printf("\n\n Premere Invio per continuare...");
scanf("%c", &invio);
}
```

Listato 7.9 Gestione di una sequenza con l'uso di funzioni per immissione, ordinamento, ricerca completa, ricerca binaria e visualizzazione