4.1 Array

Quando si ha la necessità di trattare un insieme omogeneo di dati esiste una soluzione diversa da quella di utilizzare tante variabili dello stesso tipo: definire un *array*, ovvero una variabile strutturata dove è possibile memorizzare più valori tutti dello stesso tipo. Intuitivamente, un array monodimensionale o *vettore* può essere immaginato come un contenitore suddiviso in tanti scomparti quanti sono i dati che vi si vogliono memorizzare. Ognuno di questi scomparti, detti *elementi* del vettore, contiene un unico dato ed è individuato da un numero progressivo, detto *indice*, che specifica la posizione dell'elemento all'interno del vettore stesso. L'indice può assumere valori interi da zero al numero totale di elementi meno 1. L'indice di base dell'array è sempre zero. Il numero complessivo degli elementi del vettore viene detto *lunghezza*.

In Figura 4.1, per esempio, l'elemento di indice 2 o, più brevemente, il terzo elemento del vettore contiene il carattere `Q`, il carattere `S` è contenuto nell'elemento di indice 0 e il vettore ha lunghezza 4.

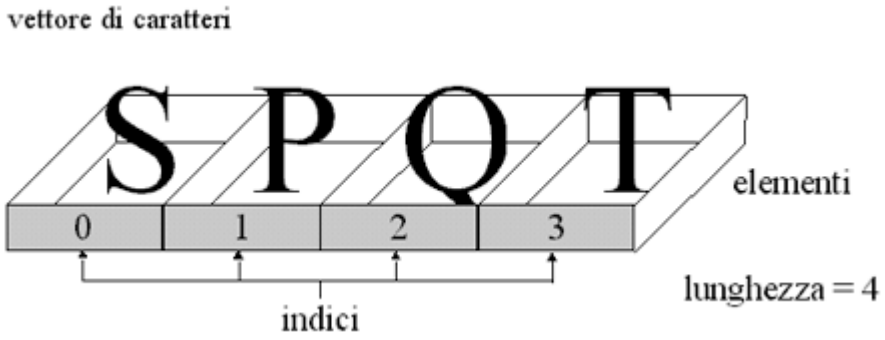

Figura 4.1 Rappresentazione intuitiva di un vettore

In definitiva, un vettore è una struttura di dati composta da un numero determinato di elementi tutti dello stesso tipo, ognuno dei quali è individuato da un indice specifico. È ora chiaro perché i vettori si dicano *variabili strutturate* mentre all'opposto tutte le variabili semplici siano anche dette *non strutturate*. Il tipo dei dati contenuti nel vettore viene detto *tipo del vettore*, ovvero si dice che il vettore è di quel particolare tipo.

Dunque per il vettore, come per qualsiasi altra variabile, devono essere definiti il nome e il tipo; inoltre si deve esplicitarne la lunghezza, cioè il numero di elementi che lo compongono. Una scrittura possibile è perciò la seguente:

```
int a[6];
```

Come sempre in C, prima deve essere dichiarato il tipo (nell'esempio `int`), poi il nome della variabile (`a`), successivamente – tra parentesi quadre – il numero degli elementi (`6`) che dev'essere un intero positivo. Questa dichiarazione permette di riservare in memoria centrale uno spazio strutturato come in Figura 4.2.

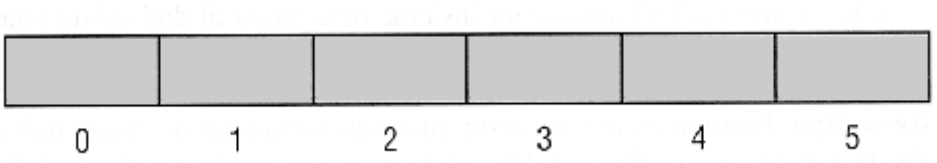

Figura 4.2 Struttura dell'array `a[6]`

Per accedere a un singolo elemento di `a` si deve specificare il nome del vettore seguito dall'indice dell'elemento posto tra parentesi quadre. L'array `a` è composto da sei elementi e l'indice può quindi assumere i valori: 0, 1, 2, 3, 4, e 5. Le istruzioni

```
a[0] = 71;
     a[1] = 4;
```

assegnano al primo elemento del vettore `a` il valore 71 e al secondo 4. Se `b` è una variabile intera (cioè dello stesso tipo del vettore), è possibile assegnare il suo valore a un elemento di `a` e viceversa:

```
a[3] = b;
```

In generale un singolo elemento dell'array può essere utilizzato esattamente come una variabile semplice. Nell'espressione

```
b = b + a[0] * a[5];
```

il valore di `b` è sommato al prodotto tra il primo e il sesto elemento di `a` e il risultato è assegnato a `b`.

Spesso l'array viene trattato all'interno di iterazioni; infatti risulta semplice far riferimento a suoi elementi incrementando ciclicamente il valore di una variabile intera e utilizzandola come indice. Consideriamo un'iterazione di esempio:

```
/* Inizializzazione dell'array */
     for(i=0; i<=5; i++) {
       printf("Inser. intero: ");
```

```
    scanf("%d", &a[i]);
  }
```

L'indice `i` dell'array `a` è inizializzato a 0 (si veda il Paragrafo 4.3) e assume a ogni iterazione successiva i valori 1, 2, 3, 4, 5. Il blocco del `for` richiede all'utente l'immissione di sei valori che vengono assegnati sequenzialmente, mediante l'istruzione `scanf`, agli elementi del vettore. Se quindi vengono inseriti in sequenza i valori 9, 18, 7, 15, 21 e 11, dopo l'esecuzione del ciclo il vettore si presenterà in memoria come in Figura 4.3.

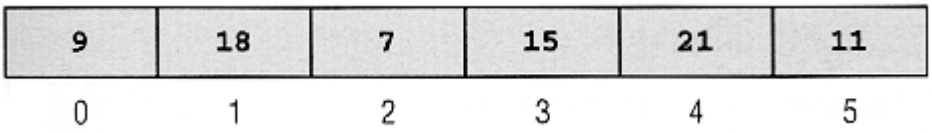

Figura 4.3 L'array `a[6]` dopo la sua inizializzazione

Anche per ricercare all'interno dell'array valori che soddisfano certe condizioni si utilizzano abitualmente i cicli:

```
/* Ricerca del maggiore */
   max = a[0];
   for(i=1; i<=5; i++)
     if(a[i]>max) max = a[i];
```

L'esempio premette di determinare il maggiore degli elementi dell'array `a`: la variabile `max` viene inizializzata al valore del primo elemento del vettore, quello con indice zero. Successivamente ogni ulteriore elemento viene confrontato con `max`: se risulta essere maggiore, il suo valore viene assegnato a `max`. Il ciclo deve comprendere dunque tutti gli elementi del vettore meno il primo, perciò l'indice `i` assume valori che vanno da 1 a 5.

Nel Listato 4.1 è riportato un programma che richiede all'utente l'immissione del punteggio raggiunto da sei studenti, li memorizza nel vettore `voti` e ne determina il maggiore, il minore e la media.

```
/* Memorizza in un array di interi il punteggio raggiunto da sei
   studenti e ne determina il maggiore, il minore e la media    */

#include <stdio.h>

main()
{
int voti[6];
int i, max, min;
float media;

printf("VOTI STUDENTI\n\n");
/* Immissione voti */
for(i=0; i<=5; i++) {
  printf("Voto %d° studente: ", i+1);
  scanf("%d", &voti[i]);
}

/* Ricerca del maggiore */
max = voti[0];
for(i=1; i<=5; i++)
  if(voti[i]>max)
    max = voti[i];

/* Ricerca del minore */
min = voti[0];
for(i=1; i<=5; i++)
  if(voti[i]<min)
    min = voti[i];
```

```
/* Calcolo della media  */
media = voti[0];
for(i=1; i<=5; i++)
  media = media + voti[i];
media = media/6;

printf("Maggiore: %d\n", max);
printf("Minore: %d\n", min);
printf("Media: %f\n", media);
}
```

Listato 4.1 Esempio di utilizzo di una variabile array

Si noti che la richiesta dei voti all'utente viene fatta evidenziando il numero d'ordine che corrisponde al valore dell'indice aumentato di una unità.

```
printf("Voto %d° studente: ", i+1);
```

Alla prima iterazione appare sullo schermo:

```
Voto 1° studente:
```

Considerazioni analoghe a quelle fatte per il calcolo del maggiore valgono per il minimo e la media.

Nella pratica la memorizzazione in un vettore ha senso quando i valori debbono essere utilizzati più volte, come nel caso precedente. Nell'esempio, comunque, i calcoli potevano essere effettuati all'interno della stessa iterazione con un notevole risparmio di tempo di esecuzione:

```
/* Ricerca maggiore, minore  e media */
     max = voti[0];
     min = voti[0];
     media = voti[0];
     for(i = 0; i <= 5; i++) {
       if(voti[i] > max)
         max = voti[i];
       if(voti[i] < min)
         min = voti[i];
       media = media+voti[i];
     }
     media = media / 6;
```

✓ **NOTA**

È importante ricordare che in C l'indice inferiore del vettore è zero e quello superiore è uguale al numero di elementi meno 1: se si desidera un array di 100 si dichiara

```
voti[100];
```

ma si deve tenere presente che l'indice assume valori da 0 a 99.

Uno degli errori più ricorrenti nella programmazione in C è lo sconfinamento dei limiti degli array. Non sempre è semplice rintracciare tali errori poiché i compilatori non li segnalano. I controlli necessari in questo senso sono a carico del programmatore.

Dopo la dichiarazione del tipo possono essere presenti i nomi di più variabili di quel tipo, semplici o strutturate. Per esempio

```
int i, voti[6], max, min, somma;
```
dichiara le variabili intere `i`, `max`, `min`, `somma` e la variabile array di tipo intero `voti` .