4.2 Esempi di uso di array

Per determinare la destrezza di *n* concorrenti sono state predisposte due prove, entrambe con una valutazione che varia da 1 a 10; il punteggio totale di ogni concorrente è dato dalla media aritmetica dei risultati delle due prove. Si richiede la visualizzazione di una tabella che contenga su ogni linea i risultati parziali e il punteggio totale di un concorrente. Nel Listato 4.2 è mostrato il programma relativo.

```
/* Carica i punteggi di n concorrenti su due prove
   Determina la classifica                          */

#include <stdio.h>

#define MAX_CONC 1000    /* massimo numero di concorrenti */
#define MIN_PUN  1       /* punteggio minimo per ogni prova */
#define MAX_PUN  10      /* punteggio massimo per ogni prova */

main()
{
float prova1[MAX_CONC], prova2[MAX_CONC], totale[MAX_CONC];
int i, n;

do {
  printf("\nNumero concorrenti: ");
  scanf("%d", &n);
}
while(n<1 || n>MAX_CONC);

/* Per ogni concorrente, richiesta punteggio nelle due prove */
for(i=0; i<n; i++) {
  printf("\nConcorrente n.%d \n", i+1);

  do {
    printf("Prima prova: ");
    scanf("%f", &prova1[i]);
  }
  while(prova1[i]<MIN_PUN || prova1[i]>MAX_PUN);

  do {
    printf("Seconda prova: ");
    scanf("%f", &prova2[i]);
  }
  while(prova2[i]<MIN_PUN || prova2[i]>MAX_PUN);
}

/* Calcolo media per concorrente */
for(i=0; i<n; i++)
  totale[i] = (prova1[i]+prova2[i])/2;

printf("\n      CLASSIFICA\n");
for(i=0; i<n; i++)
  printf("%f  %f  %f \n", prova1[i], prova2[i], totale[i]);
}
```

Listato 4.2 Esempio di utilizzo di un array

Non conoscendo a priori il numero di concorrenti che parteciperanno alle gare si fa l'ipotesi che comunque non siano più di 1000, valore che memorizziamo nella costante `MAX_CONC`. In conseguenza di ciò definiamo di lunghezza `MAX_CONC` gli array che conterranno i risultati: `prova1`, `prova2` e `totale`. Richiediamo all'utente a tempo di esecuzione il numero effettivo dei concorrenti e verifichiamo che non sia minore di 1 e maggiore di `MAX_CONC`:

```
do {
       printf("\nNumero concorrenti: ");
       scanf("%d", &n);
     }
     while(n<MIN_PUN || n>MAX_CONC);
```

In seguito richiediamo l'introduzione dei risultati della prima e della seconda prova di ogni concorrente, controllando che tale valutazione non sia minore di 1 e maggiore di 10, nel qual caso ripetiamo la richiesta. Abbiamo memorizzato in `MIN_PUN` e `MAX_PUN` i limiti inferiore e superiore del punteggio assegnabile, in maniera che, se questi venissero modificati, basterebbe intervenire sulle loro definizioni perché il programma continui a funzionare correttamente. Infine calcoliamo il punteggio totale e lo visualizziamo ▬. Un esempio di esecuzione è mostrato in Figura 4.4.

✓ **NOTA**

Si noti che soltanto *n* elementi di ogni array vengono utilizzati veramente; quindi se, per esempio, i concorrenti sono dieci, si ha un utilizzo di memoria pari a solo il 10 per mille (valore attuale di `MAX_CONC`); in questo modo però il programma è più flessibile.

Esistono altre soluzioni più complesse che permettono di gestire la memoria dinamicamente (cioè di adattarla alle effettive esigenze del programma), anziché staticamente (cioè riservando a priori uno spazio): le vedremo in seguito ▬.

```
Numero concorrenti: 3

Concorrente n.1

Prima prova: 8   Seconda prova: 7

Concorrente n.2

Prima prova: 5   Seconda prova: 9

Concorrente n.3

Prima prova: 8   Seconda prova: 8


      CLASSIFICA

8.000000  7.000000  7.500000
5.000000  9.000000  7.000000
8.000000  8.000000  8.000000
```

Figura 4.4 Esempio di esecuzione del programma del Listato 4.2