Array di strutture (tabella)

Un array di strutture viene di solito chiamato **tabella**, perché si può schematizzare come una tabella con tante colonne quanti sono i campi della struttura e un numero di righe pari al numero delle componenti dell'array.

Per esempio, i dati e le loro strutture utili per la gestione di un una biblioteca possono essere definiti nel modo seguente:

```
struct DataOdierna {
   int giorno;
   int mese;
   int anno;
};

struct Libro {
   string titolo;
   string autori[3];
   int AnnoEdiz;
   string LuogoEdiz;
   string CasaEd;
   DataOdierna DataAcquisto;
};

Libro biblioteca[1000];
```

Campo composto da un array di 3 componenti (gli autori di un libro possono essere più di uno).

Campo avente il tipo *DataOdierna* definito in precedenza.

L'intera biblioteca è definita come un array di 1000 componenti ciascuna delle quali ha la struttura di *Libro*.

Con queste definizioni di dati, nel programma, è consentito l'uso delle seguenti istruzioni:

```
biblioteca[5].DataAcquisto.mese = 7;
biblioteca[40].AnnoEdiz = 2004;
```

La struttura dati più adatta per rappresentare le informazioni di un dipendente con nome, indirizzo, livello, stipendi nei 12 mesi dell'anno più la tredicesima è la seguente:

```
struct Persona {
   string nome;
   string indirizzo;
   int livello;
   float stipendio[13];
};
```

Il campo *stipendio* è un array.

Per rappresentare le informazioni relative a tutti i dipendenti di un'azienda si può utilizzare un array, le cui componenti sono di tipo *Persona:*

```
Persona dipendente[200];
```

ESEMPIO

Classifica di una gara campestre.

Alla fine di una gara campestre vengono memorizzati, per ogni partecipante, il numero di pettorale, il nome e il tempo impiegato espresso in ore, minuti e secondi. Si vuole la stampa dei partecipanti ordinati rispetto al tempo impiegato.
Il problema si divide in tre parti: l'acquisizione dei dati, l'ordinamento e la stampa.

1 L'acquisizione dei dati prevede la memorizzazione di questi in una tabella, cioè in un *array di strutture*: ogni riga della tabella è composta da una struttura che contiene i dati di un partecipante, vale a dire il numero di pettorale, il nome e il tempo impiegato.
A sua volta il tempo impiegato è definito da una struttura contenente i seguenti campi: ore (hh), minuti (mm), secondi (ss) e il totale secondi ottenuto dalla espressione hh*3600+mm*60+ss.

2 Per ordinare i partecipanti rispetto al tempo impiegato si utilizza l'algoritmo di *sort*.

> Il campo della struttura che viene utilizzato per fare i confronti nell'algoritmo dell'ordinamento viene detto campo **chiave** dell'ordinamento.

La chiave per l'ordinamento in questo esempio è il campo *totale secondi*: questo campo è stato introdotto per facilitare l'operazione di confronto tra i tempi impiegati.

3 La stampa dei dati consiste nel trasferire in output i dati della tabella ordinata attraverso una ripetizione con contatore che permette di trattare tutte le strutture, dalla prima all'ultima, memorizzate in tabella.

Programma C++

```
// Gara.cpp: classifica di una gara
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

   const int MAX = 100;

   // dichiarazione delle strutture
   struct t {
      int hh, mm, ss;
      int TotSecondi;
   };

   struct partec {
      int pettorale;
      string nome;
      t tempo;
   };
```

Dichiarazione di inclusione per usare il manipolatore *setw*.

Tempo trasformato in secondi.

Struttura per i dati di un partecipante.

```
// prototipi delle funzioni
int ChiediDimensione();
void CaricaTabella(partec T[], int d);
void Scambia(partec& a, partec& b);
void Ordina(partec T[], int d);
void StampaTabella(partec T[], int d);

// funzione principale
int main()
{
   int dim;              // dimensione indicata dall'utente
   partec tab[MAX];      // tabella dei partecipanti

   dim = ChiediDimensione();
   CaricaTabella(tab,dim);
   Ordina(tab,dim);
   StampaTabella(tab, dim);

   return 0;
}

// dimensione della tabella
int ChiediDimensione()
{
   int d;

   do {
      cerr << "Dimensione della tabella: ";
      cin >> d;
   } while (d<1 || d>MAX);

   return d;
} // ChiediDimensione

1 // caricamento delle componenti
void CaricaTabella(partec T[], int d)
{
   for (int i=0; i<d; i++) {
      cerr << "Numero di pettorale: ";
      cin >> T[i].pettorale;
      cerr << "Nome del concorrente: ";
      cin >> T[i].nome;
      cerr << "Ore impiegate: ";
      cin >> T[i].tempo.hh;
      cerr << "Minuti: ";
      cin >> T[i].tempo.mm;
      cerr << "Secondi: ";
      cin >> T[i].tempo.ss;
      T[i].tempo.TotSecondi =  T[i].tempo.hh*3600 +
                               T[i].tempo.mm*60 + T[i].tempo.ss;
   }
} // CaricaTabella
```

Flusso **cerr** (*standard error*) per i messaggi da inviare al video come supporto dell'input dell'utente (vedi nota alla fine del codice).

Trasforma il tempo in secondi.

```
// scambio di due variabili
void Scambia(partec& a, partec& b)
{
   partec comodo;

   comodo = a;
   a = b;
   b = comodo;
} // Scambia

// ordinamento crescente dell'array
void Ordina(partec T[], int d)
{
   for (int i=0; i<d-1; i++) {
      for (int j=i+1; j<d; j++)
         if (T[i].tempo.TotSecondi>T[j].tempo.TotSecondi)
            Scambia(T[i], T[j]);
   }
} // Ordina

// visualizza le componenti
void StampaTabella(partec T[], int d)
{
   // intestazione
   cout << setw(10) << "Pettorale"
        << setw(15) << "Nome"
        << setw(5) << "h"
        << setw(3) << "m"
        << setw(3) << "s"
        << setw(10) << "arrivo"
        << endl;
   // elenco partecipanti
   for (int i=0; i<d; i++)
      cout << setw(10) << T[i].pettorale
           << setw(15) << T[i].nome
           << setw(5) << T[i].tempo.hh
           << setw(3) << T[i].tempo.mm
           << setw(3) << T[i].tempo.ss
           << setw(10) << i+1
           << endl;
} // StampaTabella
```

2

3

Scambio di due strutture passate per referenza.

Ordinamento crescente sui tempi essendo una gara di velocità.

Manipolatore *setw* per visualizzare sulla stessa riga le intestazioni e i dati, all'interno di 6 posizioni di stampa.

Il programma precedente utilizza il flusso **cerr** (*standard error*) per i messaggi da inviare al video come supporto dell'input dell'utente, e il flusso **cout** (*standard output*) per l'output dei risultati dell'elaborazione.
In questo modo, in fase di testing del programma, o nel caso non fosse disponibile una stampante, è possibile effettuare una **ridirezione dell'output** dalla linea comandi che lancia l'esecuzione del programma:

```
C:\esercizi>Gara >elenco.txt
```

Volendo invece stampare l'output su carta, occorre fare una ridirezione dell'output verso la stampante **lpt1**, con la seguente linea comandi:

```
C:\esercizi>Gara >lpt1:
```