Capitolo 10 - Strutture

Strutture

- In molte situazioni, una variabile non è sufficiente per descrivere un oggetto.
  - Ad esempio, una posizione sul piano cartesiano è identificata da due coordinate, e la si può quindi rappresentare con due variabili:
    - La coordinata orizzontale x.
    - La coordinata verticale y.
- Il C consente di riunire queste due variabili in una sola struttura, in modo da poterle trattare insieme.

3

Definizione di strutture

- Ecco come potrebbe essere definita la struttura che descrive una posizione:

```
struct posiz {
    double x;
    double y;
};
```

  - struct indica che si vuol definire una struttura di nome posiz
    - La prima componente della struttura è la variabile x, di tipo double
    - La seconda componente della stessa struttura è la variabile y
  - Le due componenti x ed y formano la struttura di tipo struct posiz

4

- Le componenti (members) di una struttura possono essere di qualunque tipo, anche array
- Una struttura può (entro certi limiti) essere considerata come un nuovo tipo
  - E' quindi possibile dichiarare variabili di questo tipo.
  - Ad esempio, ecco come dichiarare due variabili di tipo struct posiz:

```
struct posiz p1;  /*primo punto*/
struct posiz p2;  /*secondo punto*/
```

  - Dopo questa dichiarazione, p1 e p2 sono due nuove variabili
    - Ciascuna di queste è però formata da due distinte componenti: x e y

5

Usare le strutture

- Data una struttura, è ovviamente necessario poter accedere a ciascuna delle componenti
  - Questo si ottiene precedendo il nome della componente desiderata con un punto
  - Se p1 è una variabile di tipo struct posiz, formata quindi dalle due componenti x e y, l'istruzione:

    p1.x = 13.2;

    assegna il valore 13.2 alla componente x di p1
  - Una componente di struttura può quindi essere usata come una normale variabile
  - Eventuali variabili comuni aventi lo stesso nome delle componenti di una struttura (es. x,y) rimangono comunque distinte:

    x = 3.3; /*variabile semplice*/

    p1.x = 4.77; /*componente di struttura*/
  - Lo stesso vale per componenti di altre strutture che siano state chiamate con lo stesso nome

6

- Le strutture hanno il vantaggio di poter essere copiate in blocco con una semplice assegnazione:

  p2 = p1; /*copia una struttura*/

  Le componenti di p1 vengono assegnate alle corrispondenti componenti di p2.
- Un confronto di strutture non ha senso:

  if (p2 > p1) ... /*ERRORE*/
  - Il C non può sapere in che modo confrontare le varie componenti.
- È lecito confrontare componenti di strutture:

  if (p2.x > p1.x) ...
  - dato che, come abbiamo già notato, si tratta di normali variabili.

- Operazioni valide
  - Assegnare una struttura ad una strutura dello stesso tipo
  - Ottenere l'indirizzo (`&`) di una struttura
  - Accedere alle componenti di una struttura
  - Utilizzare l'operatore `sizeof` per determinare la dimensione della struttura

Esercizi

- Creare una libreria per la gestione delle matrici
  - Utilizzare l'ambiente Dev-C++ per la creazione dei progetti
  - Creare il file di intestazione (header file) "matrici.h" contenente tutti i prototipi delle funzioni

```
void CaricaMatrice (int [][], int, int);
void VisualizzaMatrice (int [][], int, int);
```

- Creare il file sorgente "matrici.c" contenente le implementazioni delle funzioni (inserire il file matrici.h nell'intestazione)

```
#include "matrici.h"
void CaricaMatrice (int matrice [][], int righe, int colonne)
{
 ...
}
```

- Testare la libreria creando un file "GestioneMatrici.c" che utilizza le funzioni definite nella libreria

```
#include "matrici.h"
int main() {
int matrice A[5][3];
...
CaricaMatrice (A, 5, 3);
...
}
```

- Definire una struttura per rappresentare un punto in uno spazio tridimensionale ed implementare:
  - una funzione per il calcolo della distanza euclidea tra due punti
  - una funzione che restituisca la somma tra due punti come somma delle coordinate dei singoli punti
  - una funzione che restituisca il punto più lontano dall'origine contenuto in un vettore di punti
  - una funzione che restituisca il punto più vicino dall'origine contenuto in un vettore di punti