Le strutture

In C i dati si dividono essenzialmente in 3 tipi

- Tipi scalari: dati atomici o elementari
  - Tipi interi: char e int (short e long, signed e unsigned)
  - Tipi reali: float e double (con la variante long)
  - Enumerazioni
- Tipi strutturati: dati non atomici o non elementari
  - Vettori
  - Strutture
  - Union
  - Puntatori
  - Funzioni
- Void

Definizione di strutture: tipi di dati composti definibili dal programmatore sulla base di altri tipi di dati atomici o composti. Per definire una struttura in C si usa il costrutto **struct**

Vediamo un esempio di struttura. Pensiamo ad un punto cartesiano Il punto cartesiano è descritto dalle sue coordinate, ovvero da un valore x (= coordinata delle ascisse) e da un valore y (= coordinata delle ordinate). Se volessi usare solo dati atomici (scalari) dovrei utilizzare, a seconda della precisione che voglio utilizzare, o una coppia di interi, oppure una coppia di float o double: quindi dovrei utilizzare due variabili, una variabile x ed una variabile y. Queste due variabili sono sostanzialmente separate l'una dall'altra e, al fatto che ad un determinato valore di x corrisponda un determinato valore di y, deve venire gestito dal programmatore in modo autonomo e, sostanzialmente, manuale. Se ho un solo punto da gestire può anche essere relativamente semplice farlo in questo modo, ma se si avesse un elevato numero di punti, ad es 10, si avrebbero 10 coppie di variabili da gestire, cioè 20 variabili, e ci sarebbe da tenere nota manualmente a quale variabile x corrisponde quale variabile y, per tutti i punti in questione. La struct ci permette di definire un tipo di dato tale per cui queste due coordinate (cioè queste due variabili x ed y) vengono accoppiate fra di loro, eliminando la possibilità

al programmatore di gestirle in modo più semplice.
Vediamo l'uso generale del costrutto struct ed un esempio

```
struct struct_name
{
    DataType member1_name;
    Data Type member2_name;
    Data Type member3_name;
    (...)
}

struct struct_name var_name;
```

```
struct point
{
    int x;
    int y;
}

struct point p;
```

Scriviamo la keyword "struct" seguita dal nome della struttura che andiamo a definire; subito dopo, racchiusi tra parentesi graffe, elenchiamo tutti gli elementi che andranno a costituire questa struttura; gli elementi che andiamo ad elencare non sono altro che i dati (atomici o compositi) che vogliamo vengano associati fra di loro nella struttura che stiamo definendo, e possono essere in numero qualunque. Per ogni elemento definiamo il Tipo ed il nome.
Nella parte destra vediamo un esempio di struttura per un punto cartesiano: definiamo la struttura che chiamiamo "point", poi nella struttura associamo fra di loro due dati atomici di tipo int, x ed y.
A questo punto ho definito un nuovo tipo di variabile, cioè una variabile di tipo "point", che contiene al suo interno due valori interi. Quando voglio utilizzare una variabile di tipo "point", dovrò semplicemente utilizzare "struct point" come se fosse un tipo di dato, quindi farò una dichiarazione come per qualsiasi altra variabile: tipo di dato + nome della variabile. Nell'esempio, per usare un tipo di dato "point" che voglio chiamare "p" scriverò semplicemente "struct point p;"; a questo punto ho a disposizione una variabile p, definita come ho specificato nella struttura, cioè in questo caso un contenitore per due valori interi, il primo che si chiama "x", il secondo che si chiama "y"; questi due dati interi sono accoppiati tra di loro nella mia variabile "p".

Vediamo, dopo averla definita, come posso lavorare con una struttura, sempre riferendoci all'esempio del punto cartesiano visto sin qui.

```
struct point
{
  int x;
  int y;
}
```

Definizione della struttura di tipo "point" come contenitore per due valori interi x ed y

```
struct point p;
```

Dichiarazione di una variabile p di tipo point

```
p.x = 5;
p.y = 6;
```

Assegnamento di un valore ai due interi x ed y che sono associati nella variabile p (struttura di tipo point)

Quando si vuole accedere ad una delle componenti che sono associate fra di loro all'interno di una struttura si usa il costrutto "nome_della_struttura . nome_della_componente". Nell'esempio, per fare un assegnamento alla componente "x" della variabile struttura di tipo point "p" scriverò semplicemente "p.x = 5;"; analogamente, per fare un assegnamento alla componente y, scriverò "p.y = 6;". Questa stessa sintassi si utilizza anche per usare le componenti di una struttura: ad esempio, se volessi confrontare il valore della componente x di p e prendere una decisione in base al fatto che tale componente sia maggiore di 0, scriverò

```
if (p.x > 0)
```

Al momento abbiamo definito una struttura: ogni volta che volessi usare una variabile del tipo definito nella struttura, dovrei scrivere "struct nome_struttura nome_variabile;".
Il linguaggio C ci permette di essere anche un po' più pigri e di definire una struttura, ovvero un nuovo tipo di dato, in modo che possa venire richiamato in modo più snello, ovvero utilizzando semplicemente il nome della struttura che abbiamo definito ogni volta che vogliamo dichiarare una variabile di quel tipo, sarà dove far precedere il nome della struttura dalla keyword "struct".

Vediamo come possiamo riscrivere il programma visto precedentemente in questo modo; dobbiamo in questo caso utilizzare la keyword "typedef"

```
typedef struct
{
  int x;
  int y;
} point
```

Definizione del tipo di dato "point" come una struttura che contiene due valori interi x ed y

```
point p;
```

Dichiarazione di una variabile "p" di tipo point

```
p.x = 5;
p.y = 6;
```

Assegnamento di un valore ai due interi x ed y che sono associati fra di loro nella variabile "p" (struttura di tipo "point")

Il comando "typedef" permette di definire un nuovo tipo di dato (che non deve necessariamente essere una struttura); in questo esempio non ho fatto altro che scrivere "typedef" seguito dal tipo di dato che voglio andare a definire (in questo caso una struttura), quindi, esattamente come fatto in precedenza, elenco tra due parentesi graffe le componenti della struttura ("tipo della componente + nome della componente"), quindi, dopo la graffa chiusa, scriverò il nome con il quale voglio che venga identificato questo nuovo tipo di dato che ho creato (in questo caso "point"). Da qui in poi avrò a disposizione "point" come tipo di dato, quindi quando mi servirà una variabile di questo tipo, la dichiarerò semplicemente come sempre fatto: "tipo variabile + nome variabile", quindi in questo caso "point p;", senza più bisogno di far precedere "point" dalla keyword "struct" come invece avrei dovuto fare con la sintassi del primo esempio di struttura visto. L'assegnamento di un valore alle due componenti di p ed eventualmente ogni altro uso di tali componenti è assolutamente identico a quanto già visto in precedenza.