22. Typedef, Struct ed Union

Corso di Algoritmi e Linguaggi di Programmazione Python/C - Angelo Cardellicchio

Angelo Cardellicchio
angelo.cardellicchio@stiima.cnr.it

Outline

- Definire nuovi tipi in C
- Usare le **`struct`** in C
- Usare le **`union`** in C

Definire nuovi tipi in C

- Finora abbiamo visto variabili esclusivamente di tipo **primitivo**
  - *Ciò significa che abbiamo trattato soltanto dati interi, decimali e sotto forma di carattere*
  - *Limitarsi a questi tipi sarebbe alquanto restrittivo!*
- In C (ed in tutti gli altri linguaggi) è possibile definire dei **tipi** personalizzati dall'utente
- Questo avviene mediante la parola chiave **`typedef`**
- Ad esempio, possiamo dichiarare un tipo **`int_pointer`** che rappresenta le variabili di tipo puntatore ad intero:

```
typedef int* int_pointer;
int a = 10;
int_pointer ip = &a;
```

Usare le `struct` in C (1)

- La vera potenza di typedef si esprime però con le struct e le union
- Abbiamo definito infatti le struct come dei **contenitori** per dati eterogenei
- Ad esempio, la struct `studente` può essere definita come segue:

```c
// Definizione di una struct studente
struct studente {
    char nome[32];
    char cognome[32];
    int matricola;
};
```

Usare le `struct` in C (2)

- Possiamo quindi usarla per creare una variabile strutturata corrispondente ad uno studente (o, in altre parole, un'**istanza** di `studente`):

```c
struct studente tizio;
```

- L'istanza può essere anche inizializzata:

```c
struct studente caio = {
    .nome = "Caio",
    .cognome = "De Caius",
    .matricola = 123456
};
```

- Notiamo che per accedere alle singole proprietà della `struct` usiamo l'operatore **punto**

Usare le **`struct`** in C (3)

- Possiamo anche definire un nuovo tipo usando **`typedef`**

```c
typedef struct studente {
    char nome[32];
    char cognome[32];
    int matricola;
} STUDENTE;

STUDENTE tizio;
```

- È importante notare come il nome dato al tipo debba essere specificato immediatamente dopo la definizione della **`struct`** (in questo caso è **`STUDENTE`**)

Usare le `struct` in C (4)

- Esistono anche i puntatori alle **`struct`**

```
STUDENTE* tp = &tizio;
```

- In questo caso, per accedere alla singola proprietà, si usa la **notazione infissa**, mediante l'operatore **`->`**:

```
char nome_estratto[32] = tp->nome;
```

Usare le **union** in C

- La sintassi delle **union** è analoga a quella delle **struct**

```c
typedef union lettura_sensore {
    long lettura_intera;
    double lettura_reale;
} LETTURA_SENSORE;
```

- Sia dal punto di vista sintattico che da quello dell'utilizzo valgono tutte le considerazioni fatte per le **struct**; tuttavia, i casi in cui si usano sono generalmente differenti.
    - *Ricordiamo infatti che le* **struct** *sono usate per rappresentare dati complessi, mentre le* **union** *per creare variabili che possono avere diverse rappresentazioni possibili*

Domande?

42