Linguaggio C: Strutture

Valeria Cardellini

Corso di Calcolatori Elettronici
A.A. 2019/20

Struct

- Una struttura (o struct) è un tipo di dato **aggregato**
- E' una collezione finita di variabili correlate (denominate *campi* o *membri*), in genere di tipo diverso, ognuna identificata da un nome
- Sintassi

```
struct [nome_etichetta] {
    tipo1 nome_campo1;
    tipo2 nome_campo2;
    …
} [nome_variabile];
```

Struct: esempi

- Dati anagrafici di una persona

```
struct dati_anagrafici {
  char cognome[20];
  char nome[20];
  int eta;
  char codice_fiscale[15];
};
```

- Punto geometrico

```
struct punto {
  int x, y;
};
```


Struct: dichiarazione

- La dichiarazione di una struttura crea un nuovo tipo di dato
- Le variabili di tipo struct sono dichiarate nello stesso modo delle variabili di altro tipo
  - Esempio

```
struct punto p1, p2;
```

- La dichiarazione di una variabile di tipo struct può essere contestuale alla dichiarazione del tipo
  - Esempio

```
struct punto {
  int x, y;   /* x e y sono i campi, scalari di tipo int */
} p1, p2;
```

  - p1 e p2 sono due variabili di tipo punto

Operazioni su struct

- Inizializzare una variabile struct
  - Con valore costanti (tra parentesi graffe)
  - Esempio: `struct punto p1 = {2, 5};`
- Assegnare variabili di tipo struct a variabili dello stesso tipo
  - Copia del valore di tutti i campi
- Rilevare l'indirizzo di una variabile di tipo struct
- Conoscere la dimensione di una struct
  - Tramite `sizeof`
- Accedere ai campi di una struct
- Dichiarare un puntatore a struct
  - Es: `struct punto *p3;`

Accedere ai campi di struct

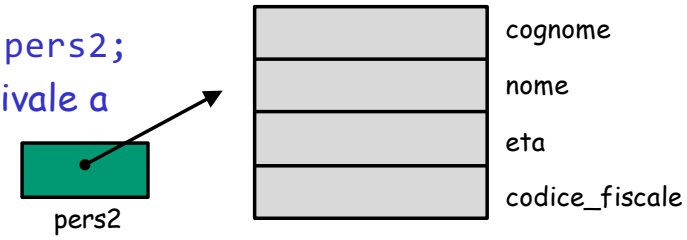
- Per accedere ai singoli campi di una variabile di tipo struct due notazioni:

1. Operatore `.` o campo di struttura

```
struct dati_anagrafici pers1;
pers1.eta = 21;
```

2. Operatore `->` o puntatore a struttura
   - L'operatore `->` permette l'accesso ai campi di una struttura per mezzo di un costrutto della forma *`<puntatore a struttura> -> <nome campo>`*
   - Esempio

```
struct dati_anagrafici *pers2;
pers2->eta = 40;      equivale a
(*pers2).eta = 40;
```



Gestione e passaggio di strutture

- A differenza degli array, il nome della variabile di tipo struct rappresenta la struttura nel suo complesso
- E' possibile:
  - Assegnare una variabile di tipo struct ad un'altra
    - ✓ Esempio: `p2 = p1;`
  - Passare ad una funzione una variabile di tipo struct
  - Far restituire ad una funzione una variabile di tipo struct
- Passaggio di parametri di tipo struct
  - Passare ad una funzione una variabile di tipo struct come parametro significa passarne una copia
  - Per passare una struttura per riferimento, occorre passare alla funzione l'indirizzo della variabile di tipo struct

typedef

- Dichiara il nome di un nuovo tipo di dato (in realtà un'abbreviazione o pseudonimo) a partire da altri tipi (scalari, aggregati, ...)
  - Sintassi

```
typedef tipoEsistente nuovoTipo;
```

- La dichiarazione di tipo è identica alla definizione di una variabile, ma è preceduta dalla clausola typedef
- Esempio:

```
typedef char string[30];
string parola;
```

  - Definisce la variabile parola di tipo `string`, cioè di tipo `char[30]`

Struct annidate

- Un campo di una struttura può a sua volta essere una struct
  - L'accesso ai campi interni richiede l'indicazione del "percorso" da seguire:

```
typedef struct {
    int giorno;
    int mese;
    int anno;
  } Data;

typedef struct {
    char *cognome;
    char *nome;
    int  eta;
    Data data_nascita;
    char *codice_fiscale;
  } dati_anagrafici;
```

Codice sorgente: struct_annidata.c