

Programmazione (imperativa)

Corso di Laurea in Informatica

Roberto Cordone

DI - Università degli Studi di Milano



Lezioni:	Lunedì 12.00 - 13.00 e 14.00 - 16.00	Mercoledì 14.00 - 17.00
Laboratorio:	Giovedì 12.00 - 13.00 e 14.00 - 17.00	
Ricevimento:	su appuntamento	
Tel.:	02 503 16235	
E-mail:	roberto.cordone@unimi.it	
Web page:	http://homes.di.unimi.it/~cordone/courses/2015-prog/2015-prog.html	

Rappresentano informazioni composte dall'**insieme di più valori concettualmente legati**

- **vettori** mono o pluridimensionali: se sono **valori omogenei associati a oggetti diversi** identificati da uno o più indici numerici (i risultati di un esame, un listino prezzi)
- **strutture**: se sono **valori eterogenei associati a uno stesso oggetto** (le misure di una stazione meteorologica, giorno mese e anno di una data)

Struttura (o **record**) è un insieme di oggetti legati fra loro, ma eterogenei, detti **membri** della struttura o **campi** del record

Una variabile di tipo struttura si dichiara specificando

- il **tipo** di ciascun membro
- il **nome** di ciascun membro
- il **nome** dell'intera variabile

```
struct {  
    tipo1 campo1;  
    tipo2 campo2;  
    ...  
} variabile;
```

La dichiarazione ha la solita struttura (*tipo variabile;*)
ma il **tipo è composto da più parole**: `struct {...}`

Applicando `sizeof` a una variabile di tipo vettore o struttura
si ottiene la memoria occupata dall'intero vettore o struttura

Dichiarazione di strutture (1)

Si può separare

- la dichiarazione della variabile (`meteo_oggi`)
- dalla dichiarazione del tipo (`dati_meteo`)

```
struct {  
    double temperatura;  
    double pressione;  
    double umidita;  
} meteo_oggi;
```

```
typedef struct {  
    double temperatura;  
    double pressione;  
    double umidita;  
} dati_meteo;  
  
dati_meteo meteo_oggi;
```

Questo evita di ripetere l'intera dichiarazione per ogni variabile

Inoltre, consente di riconoscere che le due variabili sono dello stesso tipo
(*il compilatore non saprebbe riconoscere l'uguaglianza!*)

Dichiarazione di strutture (2)

Un modo alternativo di procedere usa il **tag di struttura**

```
typedef struct {  
    double temperatura;  
    double pressione;  
    double umidita;  
} dati_meteo;
```

```
dati_meteo meteo_oggi;
```

```
struct dati_meteo {  
    double temperatura;  
    double pressione;  
    double umidita;  
};
```

```
struct dati_meteo meteo_oggi;
```

I due modi sono equivalenti (il tag è storicamente il primo)

I nomi dei campi sono locali al blocco dove sono dichiarati:
campi diversi di strutture diverse possono avere lo stesso nome

```
typedef struct {  
    long id;  
    char nome[LUNGHEZZA+1];  
    char cognome[LUNGHEZZA+1];  
} persona;
```

```
persona io, Gigi;
```

```
typedef struct {  
    int id;  
    char nome[2*LUNGHEZZA+1];  
} luogo;
```

```
luogo qui, altrove;
```

Per accedere a un campo di una struttura, si specificano la struttura e il nome del campo, separati da un punto ('.')

```
Esempio: t = meteo_oggi.temperatura;  
          meteo_oggi.pressione *= 1.1;  
          strcpy(qui.nome, "Crema");
```

L'operatore punto ha un'altissima priorità (come le parentesi quadre []) Quindi `qui.id++`; equivale a `(qui.id)++`;

I campi delle strutture possono apparire a sinistra negli assegnamenti come le variabili e gli elementi dei vettori

Annidamento (1)

Strutture e vettori possono contenere strutture e vettori ricorsivamente, con qualsiasi numero di livelli

```
typedef struct {  
    long id;  
    char nome[LUNGHEZZA+1];  
    char cognome[LUNGHEZZA+1];  
} persona;
```

```
typedef struct {  
    long matricola;  
    persona identita;  
} studente;
```

In questo modo è più facile
modularizzare il codice

- costruire funzioni che operano su strutture
- combinando funzioni che operano su sottostrutture

Annidamento (2)

Per accedere a un campo di una sottostruttura, si specificano la struttura, la sottostruttura e il campo, separati da punti ('.')

```
strcpy(studente1.identita.nome, "");
```

Per accedere a un elemento di un vettore che è campo di una struttura, si specifica la struttura, il campo vettore e la posizione

```
iniziale1 = studente1.identita.nome[0];  
iniziale2 = studente1.identita.cognome[0];
```

Per accedere a un campo di una struttura elemento di un vettore, si specifica il vettore, la posizione e il campo

```
studente classe[100];  
m = classe[12].matricola;
```

Si può applicare l'operatore di assegnamento (=) a intere strutture

```
dati_meteo meteo_ieri, meteo_oggi;
```

```
meteo_oggi = meteo_ieri;
```

equivale a

```
meteo_oggi.temperatura = meteo_ieri.temperatura;
```

```
meteo_oggi.pressione = meteo_ieri.pressione;
```

```
meteo_oggi.umidita = meteo_ieri.umidita;
```

Copia i campi della struttura a destra in quelli della struttura a sinistra

- copia i campi di tipo elementare
- copia i campi di tipo struttura, sottocampo per sottocampo
- copia i campi di tipo vettore statico (strano!)
- copia i campi di tipo puntatore, ma non duplica l'oggetto puntato (vettori dinamici!); questo può essere molto pericoloso