# LE STRUTTURE E LE UNIONI

Il linguaggio C

1



# DEFINIZIONE DI STRUTTURA

- Un **struttura (struct)** è un tipo aggregato che può contenere, a differenza di un vettore, dati eterogenei.
- In informatica questo oggetto viene comunemente definito «record».
- Ad esempio posso creare una struttura che contenga le informazioni anagrafiche di un alunno:
  - come;
  - cognome;
  - età;
  - indirizzo;
  - ecc.

# COME SI DICHIARA UNA STRUTTURA

- Per dichiarare una struttura si usa la seguente sintassi:

```
struct <NomeStruttura>
{
  <TipoCampo1> <NomeCampo1>;
  <TipoCampo2> <NomeCampo2>;
  <TipoCampo3> <NomeCampo3>;
  ...
};
```

```
struct automobile
{
  char marca[15];
  char modello[20];
  char targa[7];
  unsigned ciclindrata;
  float potenza;
};
```

- All'interno della dichiarazione:
  - non si possono inizializzare i campi;
  - si possono utilizzare tipi semplici, tipi aggregati o tipi definiti dall'utente.



# COME SI DICHIARA UNA VARIABILE A PARTIRE DA UNA STRUTTURA

- Per dichiarare una variabile di tipo «automobile», normalmente, si usa la stessa sintassi utilizzata per i tipi semplici.

```
struct automobile a1, a2, a3;
```

- In alternativa, si possono anche dichiarare le variabili insieme alla struttura, inserendo i nomi delle stesse dopo la parentesi graffa chiusa.

```
struct automobile
{
   char marca[15];
   char modello[20];
   char targa[7];
   unsigned ciclindrata;
   float potenza;
} a1, a2, a3;
```



# COME SI ACCEDE AD UN CAMPO DI UNA STRUTTURA

- Per accedere ad un campo di una struttura si utilizza l'operatore punto ".".
- Ad esempio:
  - se volessi copiare il campo «cilindrata» della variabile «a1» di tipo struttura «automobile» in una variabile «x», agirei nel modo seguente:

```
x = a1.cilindrata;
```

  - se volessi inserire da tastiera la targa dell'auto «a1» e salvarla nell'opportuno campo della variabile «a1» di tipo struttura «automobile», agirei nel modo seguente:

```
scanf("%s", a1.targa);
```

# COME SI INIZIALIZZA UNA STRUTTURA

- Una variabile di tipo struttura può essere inizializzata, contemporaneamente alla dichiarazione, con una modalità simile a quella utilizzata per i vettori, inserendo i valori di inizializzazione, separati dalla virgola, tra parentesi le graffe.

```
struct automobile a1 = {"Ferrari", "F123", "AB123CD", 5000, 700};
```



# CAMPI DI BIT

- Sono uno strumento che permette, grazie alle strutture, di ottimizzare l'utilizzo della memoria «impacchettando» variabili che necessitano pochi bit nello spazio di memoria occupato da una variabile di tipo intero.

```
struct test
{
  unsigned risp1 : 1;
  unsigned risp2 : 1;
  unsigned risp3 : 1;
  unsigned risp4 : 1;
  unsigned risp5 : 1;
  unsigned risp6 : 1;
  unsigned punteggio : 3;
};
```



# STRUCT E ARRAY

- Una struttura può contenere dati di tipo aggregato: array e/o strutture.
- Un array può essere definito a partire da un tipo «struttura».

```
struct passeggero
{
  char nome[100];
  int anni;
};

struct volo
{
  char nomeVolo[10];
  float durata;
  struct passeggero lista[100];
};

struct volo voli[200];
```

# ESEMPIO

- Calcolo della media dei voti degli studenti.

```
#include <stdio.h>
#include <stdlib.h>

struct studente
{
  char nome[30];
  unsigned voti[3];
};

int main(int argc, char *argv[])
{
  int i;
  float media;
  char str[100];
  struct studente studenti[6];
```



# ESEMPIO

```
/* Leggo da tastiera i nomi degli studenti e i relativi voti */

for(i = 0; i < 6; i++)
{
  printf("\nInserisci il nome dello studente: ");
  gets(studenti[i].nome);

  printf("\nInserisci i 3 voti dello studente: ");
  gets(str);
  sscanf(str, "%u%u%u", &studenti[i].voti[0], &studenti[i].voti[1], &studenti[i].voti[2]);
}
```

# ESEMPIO

```
  printf("\n\nMEDIE:");
  for(i = 0; i < 6; i++)
  {
    media = (studenti[i].voti[0] + studenti[i].voti[1] + studenti[i].voti[2]) / 3.0;

    printf("\n%s: %.1f", studenti[i].nome, media);
  }

  system("PAUSE");
  return 0;
}
```

# ESERCIZI

- **Esercizio 1:** scrivere un programma che, dato il nome di una serie di città (al massimo 10) e la loro temperatura minima e massima giornaliera, le visualizzi in ordine crescente rispetto alla loro temperatura media.
- Suggerimenti:
  - creare un struttura contenente i seguenti campi:
    - nomeCitta;
    - tempMin;
    - tempMax;
    - tempMedia.
  - creare un vettore di 10 elementi a partire dalla predetta struttura.



# DEFINIZIONE DI UNIONE

- Un'**unione (union)** è un'area di memoria che può contenere tipi di dato diversi in istanti differenti.
- In una unione l'indirizzo di memoria è unico, ma il numero di byte letti varia in funzione della variabile a cui ci si riferisce.
- La quantità di memoria occupata da un'unione è pari alla quantità utilizzata per memorizzare il tipo «più grande».

# COME SI DICHIARA UN'UNIONE

- Per dichiarare un'unione si usa la stessa sintassi utilizzata per le strutture, sostituendo però la parola chiave «struct» con «union»:

```
union <NomeUnion>
{
  <TipoCampo1> <NomeCampo1>;
  <TipoCampo2> <NomeCampo2>;
  <TipoCampo3> <NomeCampo3>;
  …
};
```

```
union poligono
{
  unsigned int lato;
  float perimetro;
  double area;
};
```

# STRUCT VS UNION

**struct**

```
struct poligono
{
   unsigned int lato;
   float perimetro;
   double area;
} p1;

[…]

p1.lato =  5;
p1.perimetro = 20;
p1.area = 25;

printf("Lato: %d", p1.lato);
```

Lato: 5

**union**

```
union poligono
{
   unsigned int lato;
   float perimetro;
   double area;
} p1;

[…]

p1.lato =  5;
p1.perimetro = 20;
p1.area = 25;

printf("Lato: %d", p1.lato);
```

Lato: 0 (???)