9.10 Le strutture

Corso di Laurea in Ingegneria Elettronica e Informatica

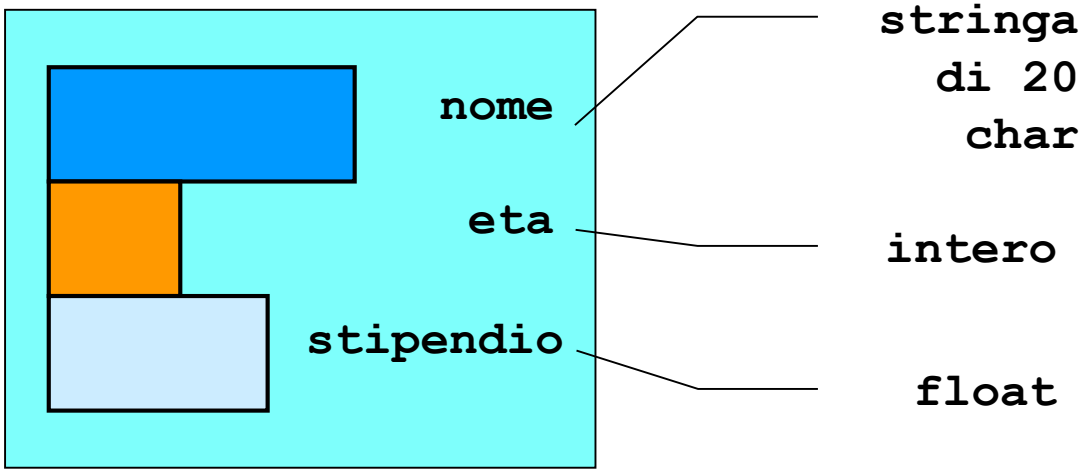
Anno accademico 2020/2021

Prof. MARCO GAVANELLI

STRUTTURE

- *Per raggruppare dati di tipo diverso ma che logicamente devono essere insieme, esistono le strutture*
- *Una* struttura *è una* collezione finita di variabili, dette campi, non necessariamente dello stesso tipo*, ognuna identificata da un* nome*.*

struct persona



STRUTTURE

Definizione di una variabile *di tipo* struttura*:*

```
struct [<etichetta>]
 {
   { <definizione-di-variabile> }
 } <nomeStruttura> ;
```

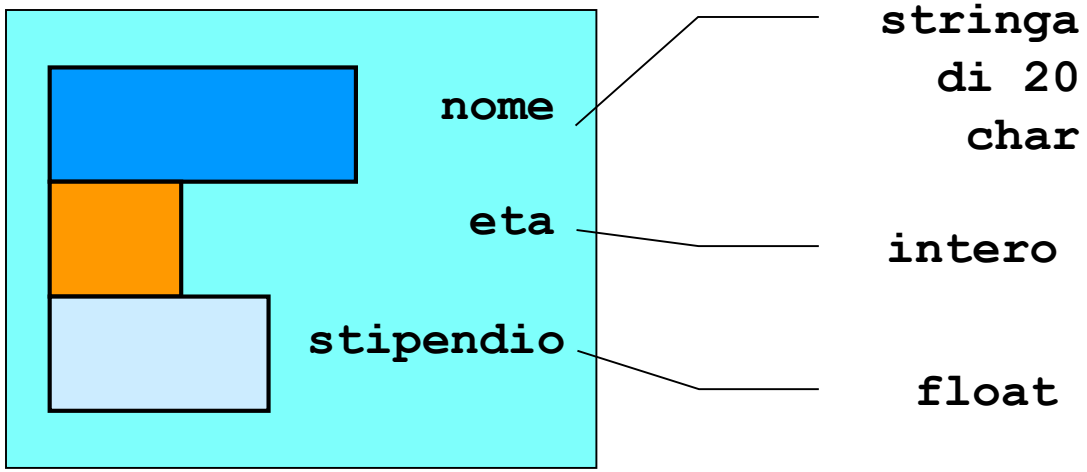
ESEMPIO

Etichetta

```
struct persona
{ char nome[20];
  int  eta;
  float stipendio;
} pers ;
```

variabile

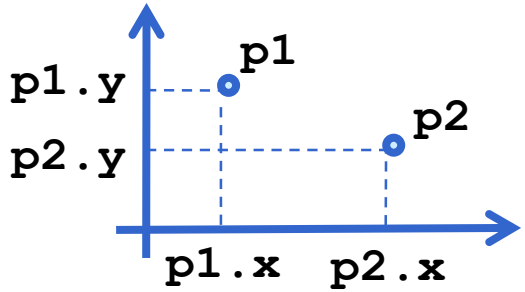
Definisce una variabile **pers** di tipo **struct persona** strutturata nel modo illustrato.



ESEMPIO

```
struct punto
{    int  x, y;
} p1, p2 ;
```

p1 *e* **p2** *sono strutture fatte ciascuna da due campi interi di nome* **x** *e* **y**




```
struct data
{    int  giorno,mese,anno;
} d ;
```

d *è costituita da tre campi interi di nome* **giorno**, **mese** *e* **anno**

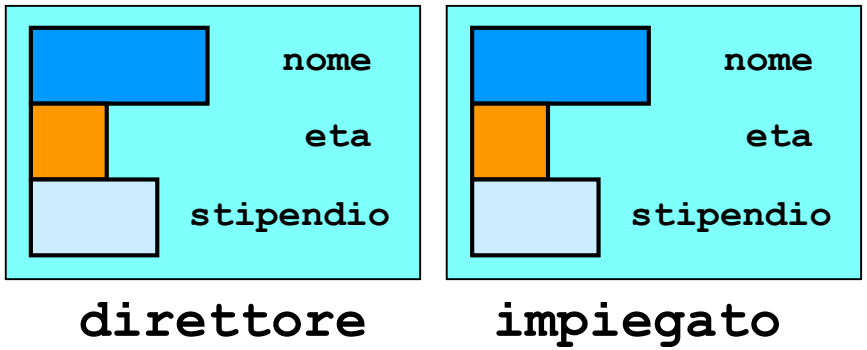
STRUTTURE: Notazione puntata

- *Una volta definita una variabile struttura,* si accede ai singoli campi mediante la notazione puntata.

Ogni campo si usa come una normale variabile del tipo corrispondente al tipo del campo.

Ad esempio:

```
struct persona
   { char nome[20];
     int eta;
     float stipendio;
   } direttore, impiegato;
```




```
impiegato.eta = 30;
if (direttore.stipendio < impiegato.stipendio)
   direttore.stipendio = 2*impiegato.stipendio;
```

Esercizio: differenza orari

- *Si leggano da tastiera due orari, ciascuno costituito da un numero intero di ore e di minuti, e li si inseriscano in opportune strutture*
- *Si calcoli il tempo trascorso tra il primo e il secondo orario (supponendo che il secondo sia successivo nel tempo)*

```
Inserisci i due orari:
6 40
10 30
differenza: 3:50
```

Strutture

9.11 Uso dell'etichetta

Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

Esempio

```
struct persona
   { char nome[20];
     int eta;
     float stipendio;
   } tecnico;
```

- *Una volta definita una variabile con una etichetta, si possono definire altre variabili usando la stessa etichetta*

```
struct persona direttore, impiegati[100];
```

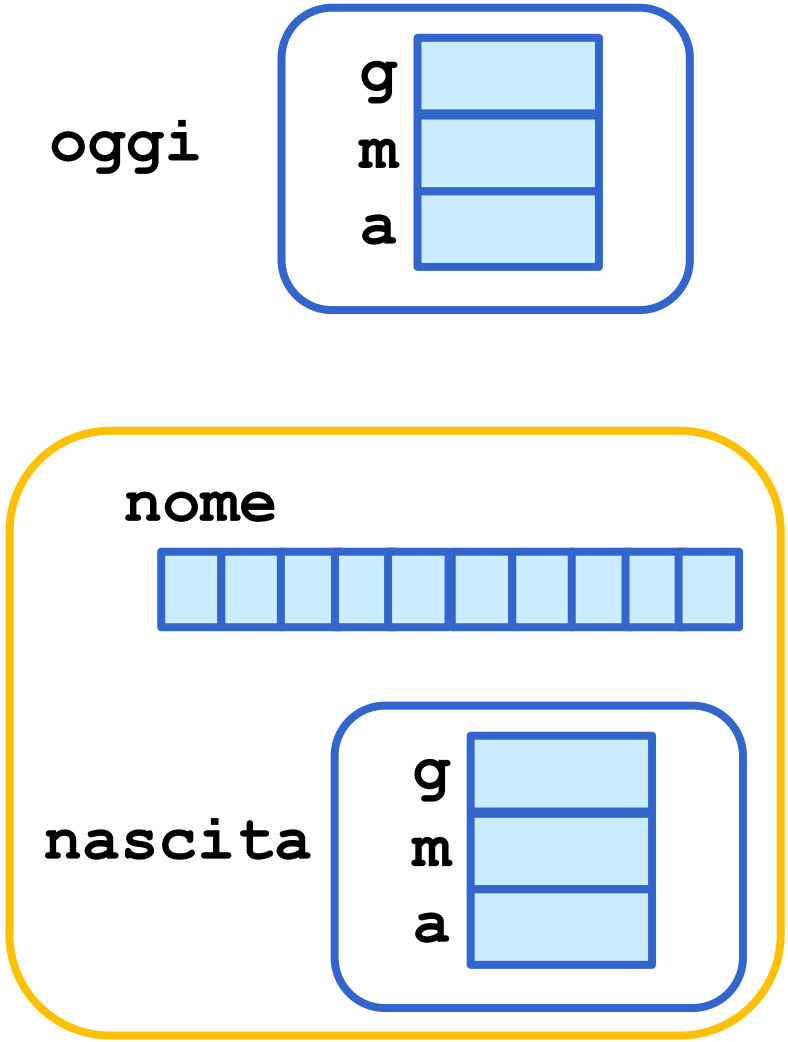
- *Non è ancora esattamente come definire un nuovo tipo (bisogna comunque mettere la parola chiave* **`struct`***)*

Esercizio: compleanno

- *Si legga da tastiera una struttura costituita da 3 campi interi contenente la data di oggi*
- *Si legga una struttura contenente il nome di una persona e la sua data di nascita*
- *Se oggi è il compleanno della persona, si stampi* "Auguri " *seguito dal nome della persona*

```
struct data
{
    int g, m, a;
} oggi;


struct persona
{
    char nome[10];
    struct data nascita;
} p;
```



Strutture

9.12 Typedef

Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

Definizione di nuovi tipi

- *Per definire un nuovo tipo di dato, si utilizza la* **typedef**

  typedef <TipoEsistente> <NuovoTipo>;

- *Es:*

```
typedef struct
  { char nome[20];
    int eta;
    float stipendio;
  } persona;
```

- *Ora posso usare* **persona** *esattamente come utilizzo i tipi predefiniti*

```
persona p, impiegati[100];
```

Esercizio: compleanno

- *Si modifichi il programma dell'esercizio precedente definendo i tipi tramite la* **`typedef`**.
- *Inoltre, l'utente dovrà inserire il nome del mese, invece del numero*

```
inserisci data di oggi:
26 ottobre 2017
inserisci dati persona:
Gigi 1 giugno 2001
non e` il tuo compleanno
```

Inizializzazione di strutture

- *Per inizializzare una variabile di tipo struttura, si elencano i valori dei campi fra parentesi graffe*
- *Es:*

```
typedef struct
{ int giorno;
  char mese[10];
  int anno;
} data;

main()
{ data Liberazione = {25,"aprile",1945};
…
}
```

Tipi e variabili

tipo

```
int a;
```

variabile

```
char c[10];
```

- *Ad un tipo possono essere associate variabili*
- *Ad un tipo non è associata alcuna area di memoria*
- *Un tipo non ha un indirizzo, né un valore. Non posso assegnargli un valore. Non posso stampare il suo valore.*

es.:

```
int = 7;
char[8] = 'A';
printf("%f",float);
```

ERRORE!!!

- *Una variabile ha sempre un tipo*
- *Ad una variabile è associata un'area di memoria*
- *essa ha un indirizzo e contiene un valore*
- *La quantità di memoria associata ad una variabile dipende dal suo tipo*

Tipi e variabili

tipo

```
typedef struct {char
nome[5]; int durata;} nota
```

variabile

```
nota n1,brano[20];
```

- *Ad un tipo possono essere associate variabili*
- *Ad un tipo non è associata alcuna area di memoria*
- *Un tipo non ha un indirizzo, né un valore. Non posso assegnargli un valore. Non posso stampare il suo valore.*

es.:

```
nota.durata = 7;
nota.nome[8] = 'A';
printf("%s",nota.nome);
```

- *Una variabile ha sempre un tipo*
- *Ad una variabile è associata un'area di memoria*
- *essa ha un indirizzo e contiene un valore*
- *La quantità di memoria associata ad una variabile dipende dal suo tipo*

ERRORE!!!

Tipi di dato strutturato

9.13 Il tipo enumerativo

Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

TIPI DEFINITI DALL'UTENTE

- ***In C, l'utente può introdurre* nuovi tipi *tramite una* definizione di tipo**
- ***La definizione associa a un identificatore* (nome del tipo) *un tipo di dato***
  - ***aumenta la leggibilità del programma***
  - ***consente di ragionare per astrazioni***
- ***Il C consente, in particolare, di:***
  - ***ridefinire tipi già esistenti***
  - ***definire dei nuovi* tipi enumerativi**
  - ***definire dei nuovi* tipi strutturati**

TIPI RIDEFINITI

- ***Un nuovo identificatore di tipo viene dichiarato identico a un tipo già esistente***
- ***Schema generale:***

```
typedef  TipoEsistente NuovoTipo ;
```

- ***Esempio***

```
typedef     int MioIntero;
MioIntero   X,Y,Z;
int         W;
```

Esempio

- *Che cosa fa questo programma?*

```
main()
{ int i, pagaOraria=…;
  int paga[7], ore[7]={…};
  for (i=0;i<=6;i++)
  {  if ((i==5) || (i==6))
          paga[i]=ore[i]*pagaOraria*1.5;
     else
          paga[i]=ore[i]*pagaOraria;
  }
}
```

TIPI ENUMERATIVI

- *Un tipo enumerativo viene specificato tramite* l'elenco dei valori *che i dati di quel tipo possono assumere.*
- *Schema generale:*

```
typedef enum
{  a1, a2, a3, ... , aN } EnumType;
```

- *Il compilatore associa a ciascun "identificativo di valore"* **a1**,…,**aN** *un* numero naturale *(0,1,...), che viene usato nella valutazione di espressioni che coinvolgono il nuovo tipo.*

TIPI ENUMERATIVI

- *Gli "identificativi di valore"* **a1, ..., aN** *sono a tutti gli effetti* delle nuove costanti.
- *Esempi*

```
typedef enum
{lun,mar,mer,gio,ven,sab,dom} Giorni;

typedef enum
{ cuori, picche, quadri, fiori} Carte;

Carte   C1, C2, C3, C4, C5;
Giorni  Giorno;
if (Giorno == dom) /* giorno festivo */
else /* giorno feriale */
```

TIPI ENUMERATIVI

- *Un "identificativo di valore" può comparire una sola volta nella definizione di un solo tipo, altrimenti si ha ambiguità.*

- *Esempio*

```
typedef enum
{lun,mar,mer,gio,ven,sab,dom} Giorni;

typedef enum
{gen,feb,mar,apr,mag,giu,lug,ago,set,ott
,nov,dic} Mesi;
```

La definizione del secondo tipo enumerativo è **scorretta**, *perché l'identificatore* `mar` *è già stato usato altrove.*

TIPI ENUMERATIVI

- *Un tipo enumerativo è* totalmente ordinato: *vale l'ordine con cui gli identificativi di valore sono stati elencati nella definizione.*

- *Esempio*

```
typedef enum
{lun,mar,mer,gio,ven,sab,dom} Giorni;
```

Data la definizione sopra,

`lun < mar` è vera

`lun >= sab` è falsa

in quanto `lun` ↔ 0, `mar` ↔ 1, `mer` ↔ 2, *etc.*

TIPI ENUMERATIVI

- *Poiché un tipo enumerativo è,* per la macchina C, *indistinguibile da un intero, è possibile in linea di principio* mischiare interi e tipi enumerativi

- *Esempio*

```
typedef enum
{lun,mar,mer,gio,ven,sab,dom} Giorni;

Giorni g;

g = 5;          /* equivale a g = sab */
```

- **È una pratica <u>da evitare</u> ovunque possibile!**

TIPI ENUMERATIVI

- ***È anche possibile specificare i valori naturali cui associare i simboli*** **`a1,...,aN`**
- **`typedef enum {lun,mar,mer,gio,ven,sab,dom} Giorni;`**
  qui, **`lun`** ↔ 0, **`mar`** ↔ 1, **`mer`** ↔ 2, *etc.*
- **`typedef enum {lun=1,mar,mer,gio,ven,sab,dom} Giorni;`**
  qui, invece, **`lun`** ↔ 1, **`mar`** ↔ 2, **`mer`** ↔ 3, *etc.*
- **`typedef enum { lun=1, mar, mer=7, gio, ven, sab, dom} Giorni;`**
  qui, infine, l'associazione è data caso per caso

IL TIPO BOOLEAN

- *Il boolean non esiste in C, ma si può facilmente definirlo:*

```
typedef enum { false, true } Boolean;
```

- *così:*

  `false` ↔ 0, `true` ↔ 1

  `false` < `true`

- *logica positiva*