

Linguaggio C

tipi di dati definiti dall'utente

Università degli Studi di Brescia

Docente: Massimiliano Giacomini

Tipi definiti dall'utente: l'idea di base

- Sono tipi non previsti direttamente dal linguaggio C: possono essere definiti dall'utente mediante l'istruzione *typedef*:

typedef <definizione di tipo>

- mediante opportuni **costruttori** di tipo
- per ogni nuovo tipo viene definito un nome (del tipo)

- Una volta definito un nuovo tipo, ad es. *voto*, la dichiarazione di una variabile viene effettuata nel solito modo:

voto votoinformatica;
↑ ↑
tipo variabile

Tipi definiti dall'utente: categorie

- Tipi semplici
 - *ridefinizione*
 - *enumerazione esplicita di valori*
- Tipi strutturati
 - *array*
 - *struct*

Ne esistono anche altri, ma noi descriveremo solo quelli indicati

Ridefinizione di tipo

- Un nuovo tipo può essere definito “uguale” ad un altro tipo:

typedef <tipoesistente> <nometipo>

- Esempio:

```
typedef int numeromaglia;  
typedef float mediagoal;
```

e poi si possono dichiarare variabili come

```
numeromaglia magliaKaka, magliaPato;  
mediagoal mgKaka, mgPato;
```

e usarle, per esempio

```
magliaKaka = 22;  
mgPato=0.35;
```

Enumerazione esplicita dei valori

- Un nuovo tipo può essere definito elencandone i possibili valori:

```
typedef enum { ..., ... ,... } <nometipo>
```

- Esempio:

```
typedef enum {Inter, Juventus, Milan} squadre;
```

```
typedef enum {Giovanni, Luca, Paolo, Matteo} nomi;
```

```
squadre squadra1=Inter, squadra2=Milan, squadraforte;
```

```
if(squadra2>squadra1)
```

```
    squadraforte=squadra2;           // questa condizione risulta vera
```

```
    else squadraforte=squadra1;
```

- In realtà, il C tratta i tipi definiti in questo modo come una ridefinizione di *int*: al primo valore è associato 0, al secondo 1, ecc.
- In tal modo, sono disponibili gli operatori ==, !=, <, ecc. ecc.

Definizione di vettori (array)

Un esempio

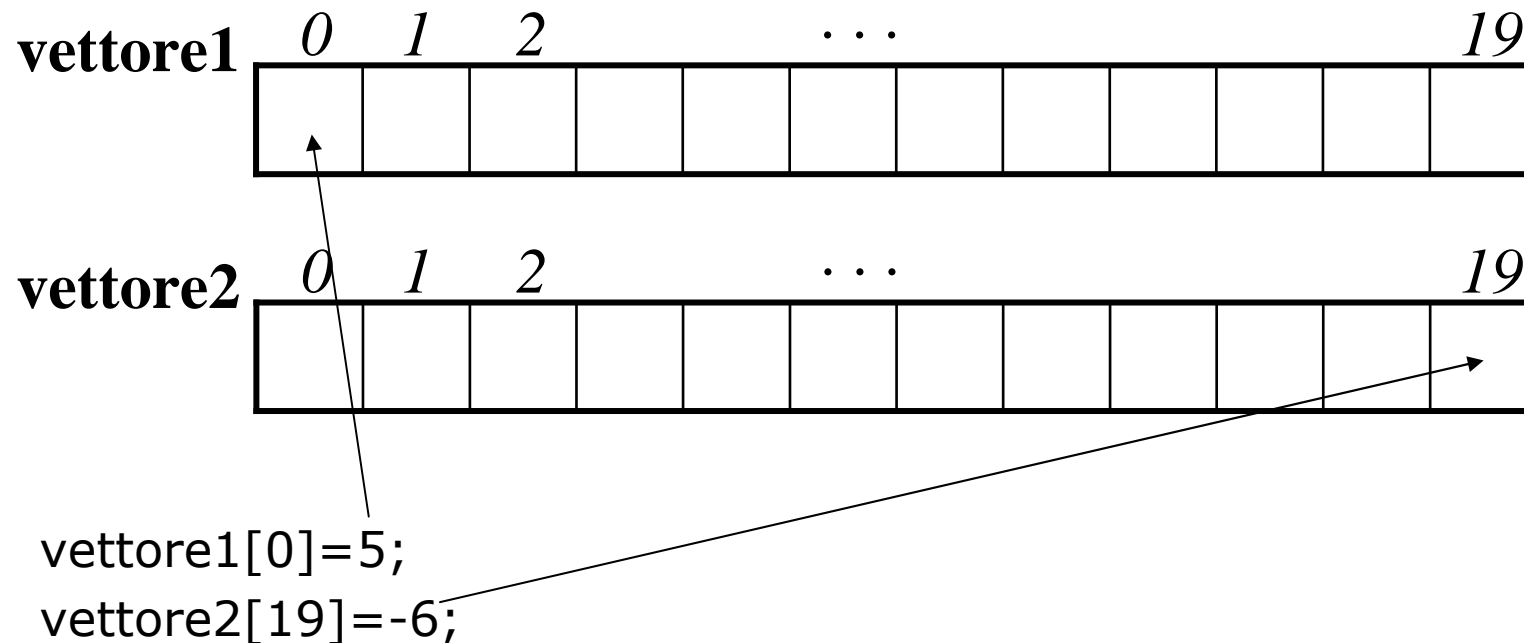
```
typedef int tipoarray[20]; ← Tipo che definisce una sequenza  
tipoarray vettore1;        di valori dello stesso tipo (int)  
tipoarray vettore2;
```



Definizione di vettori (array)

Un esempio

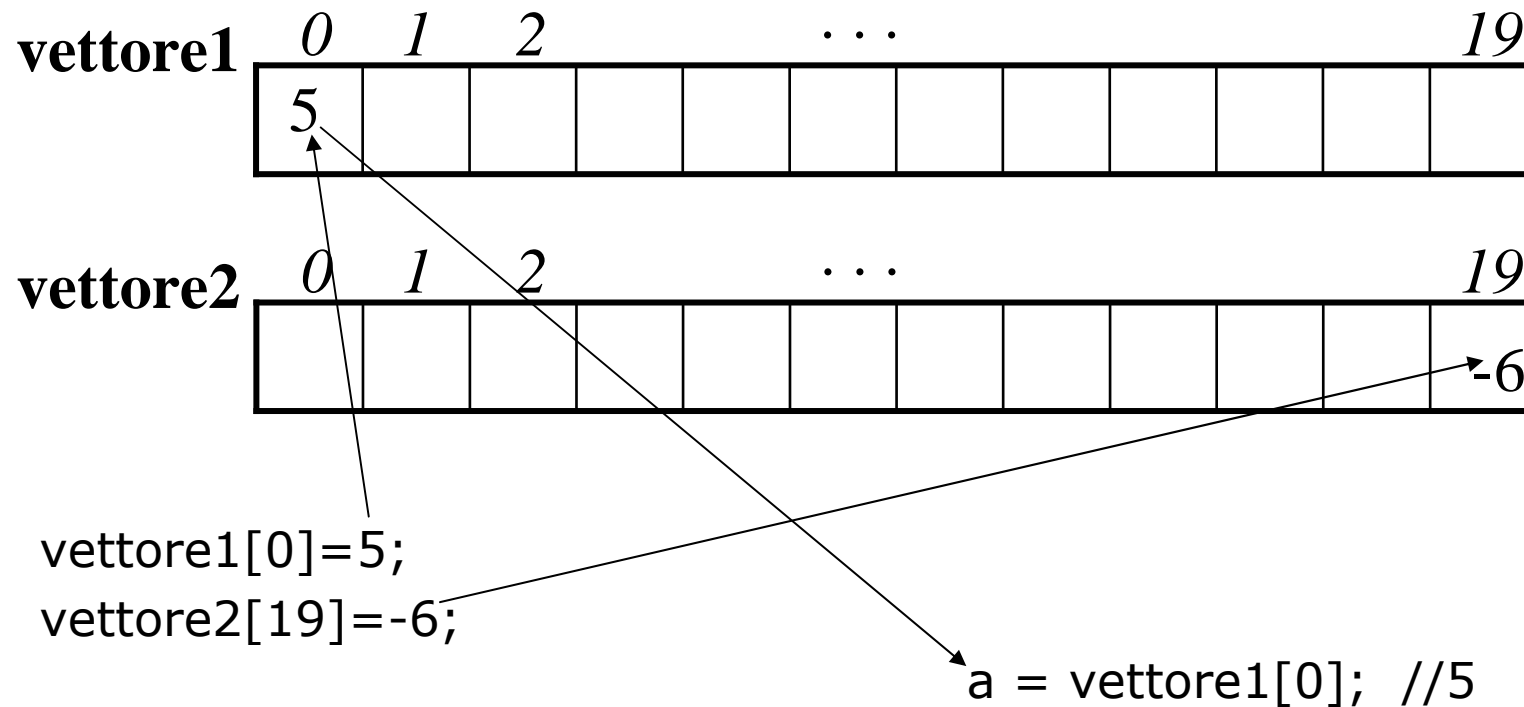
```
typedef int tipoarray[20]; ← Tipo che definisce una sequenza  
tipoarray vettore1;        di valori dello stesso tipo (int)  
tipoarray vettore2;
```



Definizione di vettori (array)

Un esempio

```
typedef int tipoarray[20]; ← Tipo che definisce una sequenza  
tipoarray vettore1;      di valori dello stesso tipo (int)  
tipoarray vettore2;
```



In generale

- Vettore: lista ordinata di elementi dello stesso tipo
- Definizione di un tipo corrispondente a un vettore:
typedef <tipo_elemento> <nometipo> [<dimensione>];
- Data una variabile *v* di tipo <nometipo>, i suoi elementi:
 - sono variabili di tipo <tipo_elemento>
 - sono identificati dalla posizione, da 0 a *dimensione-1*
 - sono denotati come *v[i]* dove *i* corrisponde alla posizione
(questa operazione si dice *indicizzazione* del vettore)
- La dimensione di un vettore è fissa: è compito del programmatore fare in modo di riferirsi a celle che compongono il vettore
(ovvero, fare in modo che $0 \leq i \leq \text{dimensione}-1$)
- E' possibile dichiarare il tipo array implicitamente nella definizione di una variabile: *<tipoelemento> <nomevariabile> [<dimensione>];*

Esempio: `int vettore1[20];`

Note

- Quando un vettore viene definito, i valori dei suoi elementi sono indefiniti
- Per definire e inizializzare un vettore con valori costanti, si può usare:

```
int vett[]={-10, 11, 4, 3};
```

```
/* equivale a      int vett[4];  
                   vett[0]=-10;  
                   vett[1]=11;  
                   vett[2]=4;  
                   vett[3]=3;    */
```

```
int vett[10]={-10, 11, 4, 3};
```

```
/* equivale a      int vett[10];  
                   vett[0]=-10;  
                   vett[1]=11;  
                   vett[2]=4;  
                   vett[3]=3;    */
```

- Il costruttore array non definisce alcun operatore specifico sul tipo di dato, ad eccezione dell'indicizzazione []:
 - sugli elementi del vettore sono definiti tutti gli operatori disponibili per il relativo tipo (es. *int*: assegnamento, confronto, ecc.)
 - sul vettore non sono disponibili operatori di assegnamento e confronto

```
vett1=vett2; //scorretto se vett1 e vett2 sono vettori
```

Utilizzo dei vettori

- Variabili capaci di memorizzare un singolo valore non sono sufficienti per tutti gli scopi
- Esempio: acquisire al più 50 valori da tastiera e stamparli in ordine inverso rispetto all'ordine con cui sono stati immessi
 - non è possibile usare 50 variabili v_1, v_2, \dots, v_{50}
 - infatti, dato che può essere stato inserito un numero n di valori, occorre un modo per riferire “la variabile i ” (per $i=1 \dots n$)
 - la soluzione è l'uso di un vettore:

```
int vett[50];
```

Esempio 1

Definire un vettore v capace di memorizzare 10 interi.
Memorizzare quindi nel vettore gli interi da 0 a 9.

Esempio 1

Definire un vettore *v* capace di memorizzare 10 interi.
Memorizzare quindi nel vettore gli interi da 0 a 9.

```
int v[10];  
for(i=0; i<10; i++)  
    v[i]=i;
```

Esempio 2

Definire un vettore v capace di memorizzare 10 interi.
Memorizzare quindi nel vettore gli interi da 1 a 10.

Esempio 2

Definire un vettore *v* capace di memorizzare 10 interi.
Memorizzare quindi nel vettore gli interi da 1 a 10.

```
int v[10];  
for(i=0; i<10; i++)  
    v[i]=i+1;
```

Esempio 3

Sviluppare un programma che acquisisca dall'utente un vettore di 10 interi. Successivamente, ne calcoli il massimo e il minimo.

Esempio 3

Sviluppare un programma che acquisisca dall'utente un vettore di 10 interi. Successivamente, ne calcoli il massimo e il minimo.

Idea

ACQUISIZIONE:

userò un ciclo per acquisire i 10 numeri (indice i da 0 a 9)

CALCOLO massimo e minimo:

userò un ciclo per confrontare due variabili max e min con tutti gli elementi del vettore.

➡ Problema: con quali valori inizializzo max e min ?
Posso inizializzarli con `vettore[0]`

Esempio 3

Sviluppare un programma che acquisisca dall'utente un vettore di 10 interi. Successivamente, ne calcoli il massimo e il minimo.

Possiamo sviluppare direttamente il codice...

```
int v[10], i, max, min;

for(i=0; i<10; i++)
    scanf("%d", &v[i]);

max=v[0];
min=v[0];
for(i=1; i<10; i++){
    if(v[i]>max)
        max=v[i];
    if(v[i]<min)
        min=v[i];
}
```

Un caso particolare di vettori: le stringhe

- Le stringhe vengono memorizzate in C in array di caratteri, il cui ultimo elemento è il carattere ‘\0’ (carattere il cui codice ASCII è 0) che indica la fine della stringa
- printf e scanf possono stampare e acquisire stringhe, mediante il carattere di conversione %s

Esempio:

```
int i;  
char stringa[20];  
scanf("%s", stringa);           // non si usa & né []  
printf("%s", stringa);          // non si usa & né []  
  
for(i=0; i<20 && stringa[i]!='\0'; i++);  
    //calcola la lunghezza della parola inserita
```

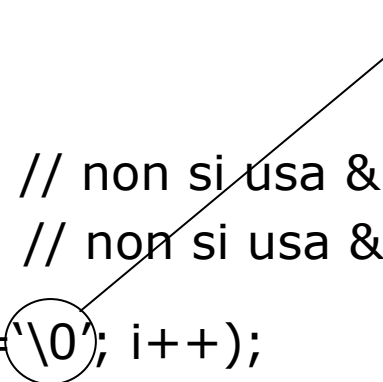
Un caso particolare di vettori: le stringhe

- Le stringhe vengono memorizzate in C in array di caratteri, il cui ultimo elemento è il carattere `'\0'` (carattere il cui codice ASCII è 0) che indica la fine della stringa
- `printf` e `scanf` possono stampare e acquisire stringhe, mediante il carattere di conversione `%s`

Esempio:

```
int i;  
char stringa[20];  
scanf("%s", stringa);           // non si usa & né []  
printf("%s", stringa);          // non si usa & né []  
for(i=0; i<20 && stringa[i]!='\0'; i++);  
    //calcola la lunghezza della parola inserita
```

0 è lo stesso!



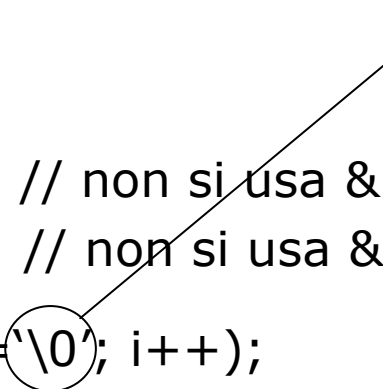
Un caso particolare di vettori: le stringhe

- Le stringhe vengono memorizzate in C in array di caratteri, il cui ultimo elemento è il carattere `'\0'` (carattere il cui codice ASCII è 0) che indica la fine della stringa
- `printf` e `scanf` possono stampare e acquisire stringhe, mediante il carattere di conversione `%s`

Esempio:

```
int i;  
char stringa[20];  
scanf("%s", stringa);           // non si usa & né []  
printf("%s", stringa);         // non si usa & né []  
for(i=0; i<20 && stringa[i]!='\0'; i++);  
    //calcola la lunghezza della parola inserita ('\0' escluso!)
```

0 è lo stesso!



- Per inizializzare un vettore con una stringa:

```
char stringa[30]="Nel mezzo del cammin"; //21 caratteri, '\0' incluso
```

Errore comune nell'uso di vettori

- Dimenticare che le posizioni sono numerate da 0 a *dimensione-1*:

```
int i;  
int vett[10];  
for(i=1; i<=10; i++) //vett[0] non è utilizzato, vett[10] non esiste  
    vett[i]=i;
```

- Il codice corretto è:

```
int i;  
int vett[10];  
for(i=0; i<10; i++)  
    vett[i]=i;
```

Definizione di tipi struct

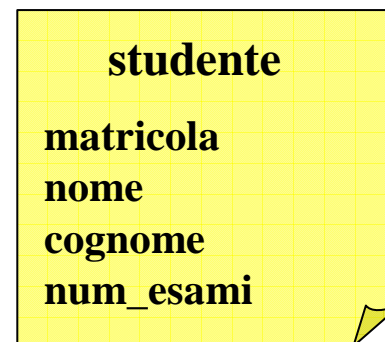
- Nei vettori: gli elementi sono tutti dello stesso tipo
- Si supponga che per ogni elemento si vogliano però memorizzare dati di tipo diverso
- Ad esempio: per ogni **studente** si vuole memorizzare *numero di matricola, nome, cognome, numero degli esami sostenuti*

Un esempio

```
typedef struct{  
    int    matricola;  
    char   nome[30];  
    char   cognome[30];  
    int    num_esami;  
} studente;
```

```
studente stud1;  
studente stud2;
```

*Definizione
del tipo
'studente'*



**Dichiarazione delle
variabili stud1 e stud2**

Un esempio

```
typedef struct{  
    int    matricola;  
    char   nome[30];  
    char   cognome[30];  
    int    num_esami;  
} studente;
```

```
studente stud1;
```

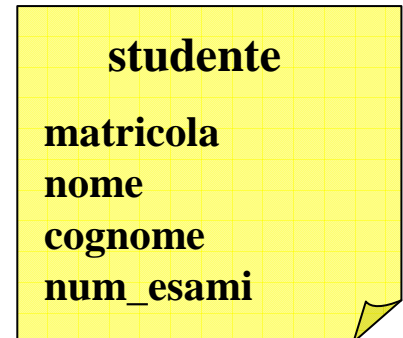
```
studente stud2;
```

```
stud1.matricola = 25891;
```

```
...
```

```
stud1.num_esami = stud2.num_esami + 1;
```

***Definizione
del tipo
'studente'***



**Dichiarazione delle
variabili stud1 e stud2**

Assegnamenti...

Definizione di tipi struct

- **Struttura** definita da:
 - insieme finito di elementi, detti *campi*
 - ogni campo:
 - > ha un *tipo specifico*
 - > è identificato da un *nome*
- Il costruttore di tipo utilizzato è *struct*
- L'unica operazione definita da *struct* è la *selezione*:
identificazione di un elemento nella forma

nome-variabile.nome-campo
- I campi si comportano come normali variabili secondo il loro
rispettivo tipo (e sono quindi disponibili i relativi operatori)

Esempio

Definire una opportuna struttura dati per memorizzare una persona, identificata dal nome, dal cognome e dall'età.

Creare un vettore che contiene i dati di 5 persone, acquisendoli da tastiera.

Quindi, stampare la somma delle età delle 5 persone.

Definizione del tipo persona e del vettore

```
typedef struct{  
    char nome[20];  
    char cognome[20];  
    int eta;  
} persona;  
  
persona vett[5];
```

Acquisizione dati da tastiera

```
for(i=0; i<5; i++){  
    printf("Persona %d:\n", i+1);  
  
    printf("Inserisci nome: ");  
    scanf("%s",vett[i].nome);  
  
    printf("Inserisci cognome: ");  
    scanf("%s",vett[i].cognome);  
  
    printf("inserisci età: ");  
    scanf("%d", &(vett[i].eta));  
}
```

Calcolo e stampa delle età

```
somma=0;  
for(i=0; i<5; i++)  
    somma+=vett[i].eta;  
printf("Somma delle età = %d\n", somma);
```

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

```
main(){
```

```
    typedef struct{
```

```
        char nome[20];
```

```
        char cognome[20];
```

```
        int eta;
```

```
    } persona;
```

```
    persona vett[5];
```

```
    int i, somma;
```

```
    for(i=0; i<5; i++){
```

```
        printf("Persona %d:\n", i+1);
```

```
        printf("Inserisci nome: ");
```

```
        scanf("%s",vett[i].nome);
```

```
        printf("Inserisci cognome: ");
```

```
        scanf("%s",vett[i].cognome);
```

```
        printf("inserisci età: ");
```

```
        scanf("%d", &(vett[i].eta));
```

```
    }
```

IL CODICE COMPLETO...

```
    somma=0;
```

```
    for(i=0; i<5; i++){
```

```
        somma+=vett[i].eta;
```

```
    printf("Somma delle età = %d\n", somma);
```

```
    system("PAUSE");
```

```
    return 0;
```

```
}
```

Esercizi proposti

- 1) Scrivere un programma che riceva in ingresso 5 interi e li memorizza in un vettore `vet_1`. Successivamente, il programma stampa i numeri in ordine inverso (p.es. se ha acquisito 3 6 9 12 1 stampa 1 12 9 6 3).
- 2) Scrivere un programma che memorizza in un vettore i dati di 5 squadre di calcio, ciascuna identificata dal nome, campionato (es. “serie a” o “serie b” ecc.) e dal numero di punti in classifica.
Stampare quindi la squadra che ha totalizzato più punti.
Cosa succede se più squadre hanno lo stesso numero di punti? Pensarci...