Università degli Studi di Ferrara

Corso di Laurea in Matematica - A.A. 2021 – 2022

Programmazione
Lezione 7 – Tipi di dato in C

Docente: Michele Ferrari - michele.ferrari@unife.it

Nelle lezioni precedenti

- Costanti in C: dati con valore fissato e strumenti utili per parametrizzare il codice
- Operatori + - * / %
- Operatori Logici && || == <= >= !=
- Funzione scanf() - acquisizione di valori dallo standard input
- Selezione in C con costrutto if
- Iterazione in C con cicli for, while-do, do-while
- Primi esercizi

In questa lezione

- Approfondimento sui tipi di dato in C
- Conversioni di tipo
- I vettori in C (array)

I Tipi di dato in C

Ricordiamo che una variabile è uno spazio di memoria identificato da un nome e a cui viene associato un **tipo**

Quando utilizziamo il nome delle variabili nelle nostre elaborazioni indichiamo che vogliamo operare sui valori che le variabili contengono

Il **tipo** determina la dimensione in memoria, i valori ammissibili e le operazioni possibili su quel dato

- In C abbiamo 5 tipi di dato principali

I Tipi di dato in C

Tipo	Keyword
Intero	int
Reale	float
Reale a doppia precisione	double
Carattere	char
Indefinito	void

Tipo “int”

Gli int (interi) occupano generalmente 2 o 4 Byte, a seconda del compilatore e dell’architettura del calcolatore.

Sono utilizzati per la rappresentazione di numeri interi con o senza segno

Alle variabili di tipo int si possono applicare (oltre al modificatore const) i modificatori short o long per aumentare o diminuire il massimo numero rappresentabile e i modificatori signed (default) o unsigned per specificare se la variabile ha il segno oppure no.

Tipo “int”

int: numeri interi con segno compresi almeno tra:
-2.147.483.648 e +2.147.483.647 (4 byte)

short int: numeri interi con segno compresi almeno tra:
-32.768 e +32.767 (2 byte)

long int: numeri interi con segno fino a 10^{19} (8 byte)

Ricordiamo: Le dimensioni massime e minime per ogni tipo sono definite nella libreria *limits.h*

Sintassi ammessa:

```
[ signed | unsigned] int
[ signed | unsigned ] short [ int ]
[ signed | unsigned ] long [ int ]
```

Tipo “int”: qualificatori di tipo

short, long, signed e unsigned sono **qualificatori di tipo**:

- short, long: condizionano lo spazio in memoria allocato*
- signed, unsigned: determina l’insieme dei valori che può assumere la variabile

** ricordiamo che la memoria, in C, non viene gestita automaticamente*

Tipo “int”: qualificatori di tipo

Tipo	Dimensione	Valori
int	4 Byte	- 2.147.483.648 + 2.147.483.647
unsigned int	4 Byte	0 4.294.967.294
short int	2 Byte	- 32.768 + 32.767
long int	8 Byte	10^{19}

→ Esempio max int

Tipo “char”

Di tutti i tipi disponibili, prestiamo particolare attenzione al tipo di dato char.

I char sono sostanzialmente dei valori interi utilizzati per rappresentare l’insieme dei caratteri disponibili sul sistema, generalmente l’insieme dei caratteri ASCII.

‘a’, ‘b’, ‘A’, ‘2’, ‘@’ ...

Nella sua forma estesa, una variabile di tipo char occupa 1 Byte, perciò è possibile rappresentare 256 (cioè 2^8) valori.

ASCII

ASCII -American Standard Code for Information Interchange, Codice Standard Americano per lo Scambio di Informazioni- è un codice per la codifica di caratteri, accettato come standard ISO.

http://it.wikipedia.org/wiki/ASCII

Tipo “char” e operatori

Abbiamo detto che una variabile di tipo char può assumere valori compresi fra 0 e 255 (nella forma estesa) quindi è possibile applicare tutti gli operatori previsti per il tipo int:

‘x’ / ‘A‘ => 120 / 65 = 1

‘R’ < ‘B’ => 82 < 65 = 0 (false)

‘x’ - 4 => 120 - 4 = 116 => ’t’

‘x’ - ‘4‘ => 120 - 52 = 68 => ‘D’

I caratteri sono ordinati secondo l’ordine lessicografico (numeri, lettere maiuscole, lettere minuscole):

0, 1, 2, ..., 9, A, B, ..., Z, a, b, ..., z

Tipo “float”

Servono a rappresentare i valori reali (con virgola), occupano 4 Byte di memoria e hanno sempre il segno.

Sono utilizzati per rappresentare l’insieme dei numeri REALI da $1.2*10^{-38}$ a $3.4*10^{38}$ con 6 cifre significative.

Sono soggetti a errori di troncamento e arrotondamento

```
float ff = 1.542342;
printf("ff / 5 = %f", ff/5); // 0,3084684
printf("ff / 7 = %f", ff/7); // 0,22033457
```

→ Esempio arrotondamento

Tipo “double”

Come le variabili di tipo float, servono a rappresentare i valori reali (con virgola), occupano 8 Byte di memoria e hanno sempre il segno.

Sono utilizzati per rappresentare l’insieme dei numeri REALI con segno da $2.2*10^{-308}$ a $1.8*10^{308}$ con 15 cifre significative .

Molti compilatori supportano anche il tipo long double per la rappresentazione di un numero reale in quadrupla precisione (10 Byte).

Espressioni omogenee ed eterogenee

Quando svolgiamo operazioni fra variabili, possiamo distinguere 2 casi:

Espressione omogenea

Espressione in cui tutti gli operandi hanno lo stesso tipo

esempio: 2 + 5

Espressione eterogenea

Espressione in cui gli operandi non hanno tutti lo stesso tipo

esempio: 2 + 6.3

Conversioni di tipo

Il compilatore C è in grado di eseguire solamente espressioni omogenee.

5 + 4

3.4 + 7.4

~~3.5 + 6~~

Quando un operatore (+, *, -, /, = ...) ha due operandi di diverso tipo, si rende necessaria una conversione ad un tipo di dato comune per render l'espressione omogenea.

Conversione implicita

Il compilatore C esegue automaticamente le conversioni di tipo che NON provocano perdita di informazione secondo regole molto rigide.

In particolare le promozioni di tipo vengo eseguite secondo la seguente gerarchia:

char → int → float → double

Conversione esplicita

Il C fornisce un costrutto sintattico per cambiare esplicitamente il tipo di una espressione:

(<nuovo_tipo>) variabile

o

(<nuovo_tipo>) espressione

Il valore di ritorno è forzatamente convertito (con eventuali perdite di informazione) al tipo nuovo_tipo

```
int an_int;
double a_double = 123.45;
an_int = (int) a_double;
```

→ Esempio conversioni

Formattatori per l’output

Formattatore	Tipo
%d	int
%ld	long int
%u	unsigned int
%f, %.*x*f	float, double
%c	char

→ Esempio formattatori

Precedenze negli operatori

Operatore
( ) [ ] . ->
Operatori unari ! ++ -- * &
* / %
+ -
< <= > >=
== !=
&&
||
: ?
= += *= ...
,

→ Esempio precedenza

La scelta multipla switch

Un altro modo per organizzare le scelte multiple è attraverso il costrutto switch-case:

```
switch (espressione)
{
  case valore1:
    // istruzioni
  break;

  case valore2:
    // istruzioni
  break;
  ...

  default:
    // istruzioni
}
```

Il risultato dell'espressione può essere solamente di tipo int oppure char.

La presenza della keyword **break** è opzionale, tuttavia, in mancanza di questa al termine del set di istruzioni di un blocco case l'esecuzione continuerebbe ai blocchi di istruzioni successivi fino ad incontrare una keyword break o la keyword default (anch'essa opzionale).

→ Esempio switch

Gli Array in C

L’Array (o vettore), in C, è una struttura dati utilizzata per memorizzare un insieme di dati di tipo omogeneo (tutti dello stesso tipo).

Esempio: array di int

5	0	34	15	12	10	0	27	8	81

Gli Array in C

Ogni elemento memorizzato nell'array è identificato da una posizione (che viene richiamata da un indice numerico)

Esempio: array di int

0	1	2	3	4	5	6	7	8	9
5	0	34	15	12	10	0	27	8	81

Gli Array in C: dichiarazione

```
<tipo> nome_array[<numero_elementi>];
```

Esempio:

```
int giorni[30];
```

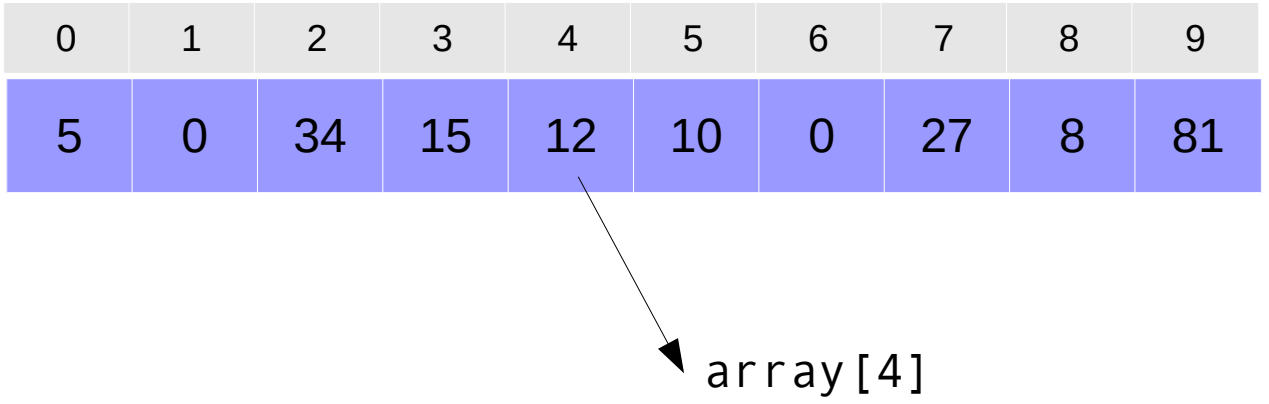
È importantissimo tenere in considerazione che il primo elemento di un array ha indice 0:

gli elementi di un array di 30 elementi sono accessibili dall'indice 0 al 29.

Gli Array in C

E’ possibile accedere ad uno specifico elemento utilizzando un **indice** che viene specificato fra parentesi quadre [ ]

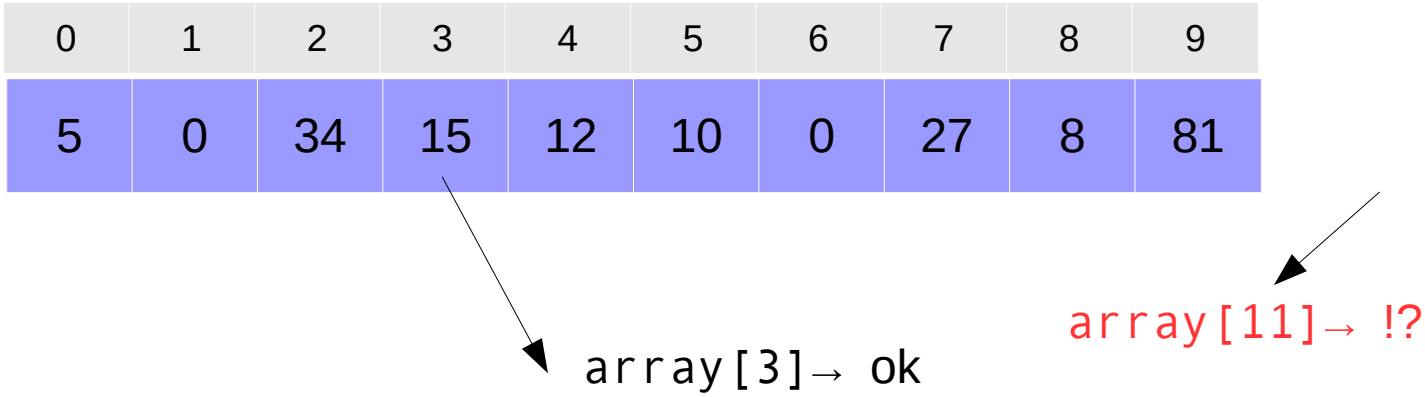
```
int array[10];
```



Gli Array in C

E’ importante ricordare che, prima di accedere alla memoria, non c’è nessun controllo sull’indice: se viene specificato un indice fuori dai limiti, il programma terminerà andando in crash

```
int array[10];
```



Esempio: come manipolare gli array

```
int main(void)
{
   int numero[10];
   int i, valore;

   // inserimento
   for (i=0; i<10; i++) {
      scanf("%d", &valore);
      numero[i] = valore;
   }

   // visualizzazione
   for (i=0; i<10; i++) {
      printf("%d", numero[i]);
   }
   return 0;
}
```

Esercizio 1

Le temperature rilevate nel corso della giornata (con intervalli di un’ora) in una determinata località sono conservate in un array.

- Scrivete un programma che richiede all’utente di inserire le temperature e le memorizzi in un array.
- Al termine dell’inserimento, il programma deve visualizzare a schermo il contenuto dell’array.

Esercizio 2

Modificate il programma precedente in modo che al termine dell'inserimento delle temperature da parte dell'utente, il programma visualizzi:

- Temperatura massima
- Temperatura minima
- Temperatura media

Grazie per l’attenzione

Riferimenti

Il corso di programmazione per il primo anno della Laurea Triennale in Matematica nasce con l'intento di unire ai principi di programmazione una conoscenza basilare di uno degli strumenti software più diffusi nell'ambito matematico: Matlab.

La prima parte del corso pertanto è una rielaborazione del programma di Programmazione per la Laurea Triennale in Informatica 15/16 (in particolare) e 16/17.

Parte del materiale originale da cui ho ricavato il percorso didattico, alcuni approfondimenti ed integrazioni possono essere trovati in:

- Lezioni di Programmazione 15/16 CdS Informatica - Giacomo Piva
- Lezioni di Programmazione 16/17 CdS Informatica - Marco Alberti