Linguaggio C

Tipi predefiniti e operatori

Università degli Studi di Brescia

Docente: Massimiliano Giacomin

RICHIAMI

char	8 bit	Valori interi compresi tra -128 e 127
int	di solito 32 bit	Valori interi (compresi tra -2,147,483,648 e +2,147,483,647)
float	di solito 32 bit	Valori razionali (numeri con la virgola)
double	di solito 64 bit	Valori razionali (numeri con la virgola)

Cosa sono gli operatori?

- Ricevono in ingresso uno o più valori di un certo tipo (operandi)
- Restituiscono un valore di un certo tipo (eventualmente diverso)

Esempio

L'operatore aritmetico +

5 + 2 ⟵ Restituisce 7

x + y ⟵ Restituisce il valore pari a x+y

- In C esistono ***operatori binari*** (due valori) ed ***operatori unari***
- Per ogni tipo operatori specifici: p.es. per *int* esiste %, per il *float* no
- Diversi operatori possono avere nome uguale:
  p.es. operatore / è diverso per *int* vs. *float*

Operatore di assegnamento

- Sintassi:

  nomevariabile = espressione

- Espressione:

  costruita a partire da variabili e/o costanti eventualmente mediante operatori.

  Ad esempio: x + (3 *y)

- Significato dell'operatore di assegnamento:

  > prima l'espressione viene valutata (viene calcolato un valore)

  > poi il valore viene assegnato alla variabile a sinistra

- Esempio

```
int x = 3;
int y;

y = (x+2)*3     // y=15
x = x+1;        // x=4
```

- Come tutti gli operatori, anche l'assegnamento restituisce un valore, ovvero il valore assegnato:

```
int x, y;
y=(x=3);        //x=3 assegna 3 a x e restituisce 3, assegnato a y:
                //x=3, y=3
```

Operatori aritmetici

- Operatori binari (già visti in precedenza):
  - `+` addizione
  - `*` moltiplicazione
  - `-` sottrazione
  - `/` divisione (diversa per tipo *int/char* o tipo *float/double*)
  - `%` resto (disponibile solo per interi *int/char*)

- Operatore unario: segno negativo `-`

- Per i tipi *int* e *char*, la divisione è intera (il risultato è di tipo *int*/*char*). Per ottenere il resto è disponibile l'operatore %.

  Esempio:

```
int dividendo=7, divisore=2,quoziente, resto;
quoziente=dividendo/divisore;                    // 3
resto=dividendo%divisore;                        // 1
```

- Per tipi *float* e *double* la divisione è diversa e restituisce un tipo *float*/*double*, ma ha lo stesso simbolo /. L'operatore resto non ha senso!

  Esempio:

```
float dividendo=7.0, divisore=2.0, quoziente;
quoziente=dividendo/divosore;                    // 3,5
```

- Quindi il simbolo ha un comportamento diverso a seconda del tipo… vediamo alcuni esempi

- Un esempio: media tra tre numeri interi

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

main(){
    int a, b, c, media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;
    printf(”La media tra %d e %d e %d fa %d\n”, a,b,c,media);
    system("pause");
}
```

Quale numero viene stampato?

- Un esempio: media tra tre numeri interi

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

main(){
    int a, b, c, media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;
    printf(”La media tra %d e %d e %d fa %d\n”, a,b,c,media);
    system("pause");
}
```

7

- Un esempio: media tra tre numeri interi

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

main(){
    float a, b, c, media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;
    printf(”La media tra %f e %f e %f fa %f\n”, a,b,c,media);
    system("pause");
}
```

Quale numero viene stampato?

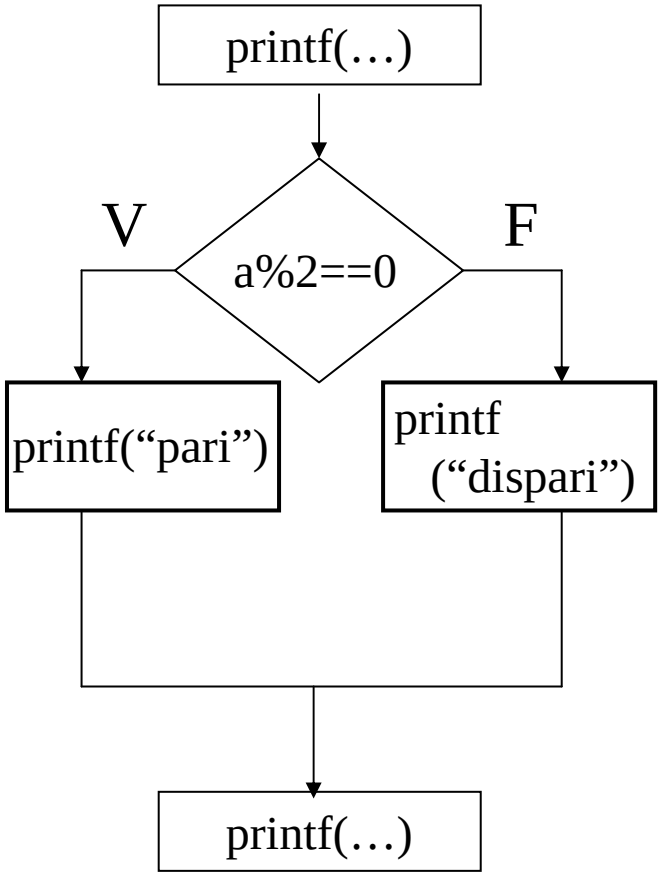
- Un esempio: media tra tre numeri interi

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

main(){
    float a, b, c, media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;
    printf("La media tra %f e %f e %f fa %f\n", a,b,c,media);
    system("pause");
}
```

7.666667

Esempio tipico di utilizzo dell'operatore resto

```
int a;

…

…

printf("la variabile a contiene un numero ");

if((a%2) ==0)
    printf("pari\n");
else
    printf("dispari\n");

printf("E' comunque un numero!\n");
```



Forme abbreviate

- `e1 op= e2` equivale a `e1 = e1 op e2`, per esempio:

```
a+=b;    //a=a+b
a*=b;    //a=a*b
```

Operatori di incremento e decremento

- ++a; //a=a+1 (come tale la valutazione dell'espressione *++a* è *a+1*, ovvero prima viene incrementata la variabile *a* poi viene valutata l'espressione)
- a++; //a=a+1, in cui però la valutazione dell'espressione *a++* è *a* (prima si valuta l'espressione poi si incrementa *a*)
- --a; //a=a-1, in cui la valutazione dell'espressione è a-1
- a--; //a=a-1, in cui la valutazione dell'espressione è a

- Esempi su forma prefissa e postfissa

```
#include<stdio.h>

main(){
    int x,y;

    x=5;
    y=x++;              // y=5, x=6
                        // equivale a y=x; x=x+1
    x=5;
    y=(x=x+1);          // x=6, y=6

    x=5;
    y=++x;              //x=6, y=6
                        //equivale a x=x+1;y=x

    …
}
```

Operatori relazionali

- Operatori su tipi numerici:

==	uguale (da non confondere con = !!!!!!!!)
<	minore
<=	minore o uguale
>	maggiore
>=	maggiore o uguale
!=	non uguale

- Confrontano due valori e restituiscono un numero:

  0 se la condizione non è verificata (“falso”)

  un numero >0 se la condizione è verificata (“vero”)

- Operatori logici:

!	NOT (operatore unario)
&&	AND (operatore binario)
||	OR (operatore binario)

- Ricevono in ingresso dei valori, interpretandoli come:

  “falso” se 0

  “vero” se diverso da 0

- Restituiscono il corrispondente valore di verità (cfr. Algebra Boole) rappresentato da un numero: 0 per “falso”, > 0 per “vero”

- Ad esempio, OR restituisce un valore > 0 se almeno uno degli operandi è ≠ 0, restituisce 0 se entrambi gli operandi sono nulli

Esempio

```
int x=5, y, z;
y = (x==5);              \\ y=1 (o comunque un valore >0)
z = (x=5);               \\ z=5
z = (x>5);               \\ z=0
x = (x=x);               \\ x inalterato
x = (x==x);              \\ x=1 (o comunque un valore >0)
x = (x!=x);              \\ x=0
z = ((x<y) && (y!=1))    \\ z=0
z = ((x<y) || (y!=1))    \\ z=1 (o comunque un valore >0)
z = !(x<y)               \\ z=0
```

ERRORE TIPICO: VERSIONE 1

```
int n;
…
…
if(2 <= n <=10)
  printf("n compreso tra 2 e 10\n");
```

La condizione è SEMPRE verificata per qualunque n!!!

ERRORE TIPICO: VERSIONE 2

```
int n;
…
…
if(-5 <= n <-2)
  printf("n compreso tra 2 e 10\n");
```

La condizione non è MAI verificata per qualunque n!!!

Conversioni implicite di tipo

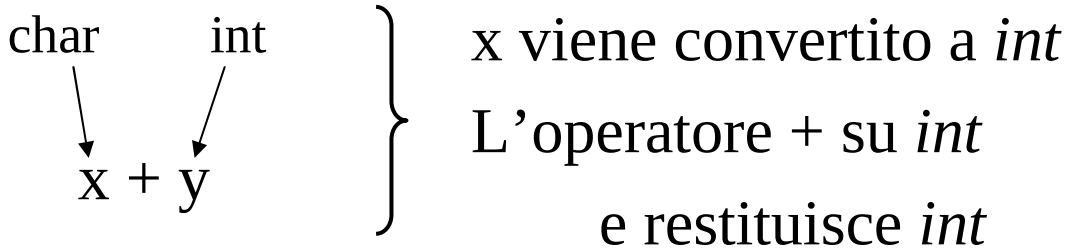
- Operatori applicati ad operandi di tipo diverso:
  regole di conversione implicita su tipi "compatibili"

Espressioni *x op y* (*op*: operatore aritmetico)

- Le regole generali sono complicate
- In genere, la conversione è al tipo "superiore":

  char → *int* → *float*

Esempio:



Operatore di di assegnamento:

- La conversione è verso il tipo della variabile assegnata
- Esempio: se *f* è *float* e *i* è *int*

```
f = i;      // valore di i convertito in un float e assegnato a f
            // (non c'è perdita di informazione)
            // es: se i=5, a f viene assegnato 5.0

i = f;      // valore di f convertito in int (perdita di informazione
            // se ad esempio f non è un intero)
            // es: se f=5.4, a i viene assegnato 5
```

- Un esempio (poco ragionevole): media tra tre numeri interi

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

main(){
    float a, b, c;
    int media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;        //valore 7.666667 troncato e assegnato
                            //a media!
    printf(”La media tra %f e %f e %f fa %d\n”, a,b,c,media);
    system("pause");
}
```

- Un esempio (poco ragionevole): media tra tre numeri interi

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

main(){
    float a, b, c;
    int media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;       //valore 7.666667 troncato e assegnato
                           //a media!
    printf("La media tra %f e %f e %f fa %d\n", a,b,c,media);
    system("pause");
}
```

7

- Un esempio: media tra tre numeri interi

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

main(){
    int a, b, c;
    float media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;        //valore intero 7 assegnato a media!
    printf("La media tra %d e %d e %d fa %f\n", a,b,c,media);
    system("pause");
}
```

- Un esempio: media tra tre numeri interi

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

main(){
    int a, b, c;
    float media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;        //valore intero 7 assegnato a media!
    printf("La media tra %d e %d e %d fa %f\n", a,b,c,media);
    system("pause");
}
```

7.000000

- Un esempio: media tra tre numeri interi

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

main(){
    int a, b;
    float c;
    float media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;        //a+b+c: conversione a float e quindi
                            //la divisione è tra float: 7.666667
                            //valore float assegnato a media!
    printf(”La media tra %d e %d e %f fa %f\n”, a,b,c,media);
    system("pause");
}
```

- Un esempio: media tra tre numeri interi

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

main(){
    int a, b;
    float c;
    float media;
    a=5;
    b=7;
    c=11;
    media=(a+b+c)/3;        //a+b+c: conversione a float e quindi
                            //la divisione è tra float: 7.666667
                            //valore float assegnato a media!
    printf("La media tra %d e %d e %f fa %f\n", a,b,c,media);
    system("pause");
}
```

7.666667

Tipi di dati semplici predefiniti:

quadro di approfondimento

Tipi di dati semplici predefiniti

1) Tipi che definiscono “variabili intere”

CHAR

[signed] char unsigned char

Spazio allocato: 1 byte
- Signed: -128 ≤ v ≤ 127
- Unsigned: 0 ≤ v ≤ 255

Tipicamente rappresentano caratteri

INT

[signed] short int [signed] int [signed] long int

unsigned short int unsigned int unsigned long int

Spazio allocato (lo stesso per signed e unsigned):
spazio(short int) <= spazio(int) <= spazio(long int)

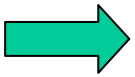
Tipicamente:
- short int: 2 byte (-32768 ≤ v ≤ 32767 o 0 ≤ v ≤ 65535)
- int e long int: 4 byte (-2,147,483,648 ≤ v ≤ 2,147,483,647 o 0 ≤ v ≤ 4,294,967,295)

Operatori su variabili int e char

=	Assegnamento (int-int o char-char)	Risultato: il valore assegnato (int o char)
+	Somma (tra int o char)	Producono risultato int o char
-	Sottrazione (tra int o char)	
*	Moltiplicazione (tra int o char)	
/	Divisione (tra int o char) con troncamento della parte frazionaria	
%	Resto (tra int o char) della divisione intera	
==	Relazione di uguaglianza (tra int o char)	Producono risultato (int o char): - 0 per “falso” - >0 per “vero”
!=	Relazione di disuguaglianza (tra int o char)	
<	Relazione “minore di” (tra int o char)	
>	Relazione “maggiore di” (tra int o char)	
<=	Relazione “minore o uguale” (tra int o char)	
>=	Relazione “maggiore o uguale” (tra int o char)	

2) Tipi che definiscono “variabili reali”

- I numeri sono rappresentati nel calcolatore in virgola mobile
- In questo caso non abbiamo le varianti signed vs. unsigned: i numeri sono tutti con segno (cfr. rappresentaz. in virgola mobile)
- I tipi sono due: FLOAT e DOUBLE, quest’ultimo ha anche la variante LONG



float double long double

Spazio allocato:
spazio(float) <= spazio(double) <= spazio(long double)

Tipicamente:
float: 4 byte
double: 8 byte
long double: spesso anche per essi 8 byte!

Operatori su variabili float e double

=	Assegnamento	Risultato: il valore assegnato (float o double)
+	Somma	Producono risultato float o double
-	Sottrazione	
*	Moltiplicazione	
/	Divisione a risultato reale	
==	Relazione di uguaglianza	Producono risultato intero: - 0 per “falso” - >0 per “vero”
!=	Relazione di disuguaglianza (tra int o char)	
<	Relazione “minore di” (tra int o char)	
>	Relazione “maggiore di” (tra int o char)	
<=	Relazione “minore o uguale” (tra int o char)	
>=	Relazione “maggiore o uguale” (tra int o char)	

Costanti numeriche e definizione di costanti

Costanti numeriche

Intere

- Per default sono di tipo *int*
- Se si aggiunge L/U si intendono di tipo *long*/*unsigned*

Esempio:

```
int a = 5;       // per default 5 è di tipo int
```

Floating point (costanti con la virgola, ad esempio 5.0)

- Per default sono di tipo *double*
- Se si aggiunge f: tipo *float*/ d:*unsigned*

Esempio:

```
double a = 5.2;       // per default 5.2 è di tipo double
```

Un esempio semplice?

```
float a;
a=0.1;
if (a==0.1)
   printf("Sì, è uguale\n");
   else printf("No, non è uguale\n");
```

PROVARE PER CREDERE…

Una semplice variante?

```
float a;
a=0.1;
if (a==0.1f)
   printf("Sì, è uguale\n");
   else printf("No, non è uguale\n");
```

PROVARE PER CREDERE…

COSA SUCCEDE NEL PRIMO CASO…

```
float a;
a=0.1;
```

in precisione doppia: 0.10000000000…

perde precisione e diventa 0.10000000149…

0.10000000149… esteso a precisione doppia

```
if (a==0.1)
   printf("Sì, è uguale\n");
   else printf("No, non è uguale\n");
```

in precisione doppia: 0.10000000000…

Costanti

- Rappresentano associazioni identificatore - costante:
  nel programma utilizzo l'identificatore al posto del valore associato
- La loro definizione è simile a quella delle variabili, premettendo la parola chiave const
- Esempio:

```
const float nofebbre = 36.6;

float temperatura;
…
if(temperatura>nofebbre)
      printf("Hai la febbre\n");
```