



Insegnamento di Elementi di informatica (6 CFU)

Il linguaggio C - Parte 2

Nadia Ranaldo
RCOST - Dipartimento di Ingegneria
Università degli Studi del Sannio

Riepilogo: istruzione di uscita (output)



printf: funzione che permette di stampare una stringa di caratteri e anche il valore di una o più variabili

• Sintassi

`printf(<stringa di controllo del formato>, <argomenti>)`

<stringa di controllo del formato>:

frase che deve essere visualizzata, comprensiva dei riferimenti ai tipi delle variabili da visualizzare

<argomenti>: variabili contenenti i valori da visualizzare

es. `printf("%d %d %d", a, b, c);`

3

Riepilogo: Dichiarazioni



```
int a,b,c;
```

Dichiarazione di tre variabili (a, b, c) di tipo intero

Consiglio: usare identificatori con un significato!

- Nelle dichiarazioni può essere anche definito un **valore iniziale**, che viene automaticamente assegnato alle posizioni di memoria

Esempio:

```
int a;
```

```
int b=5;
```

2

Esempi di printf



```
#include<stdio.h>
main()
{
    int x=3;
    int y=5;
    printf("Valore di x: %d\n", x);
    printf("Valore di y: %d\n", y);
}
```

Valore di x: 3

Valore di y: 5

```
#include<stdio.h>
main()
{
    int x=3;
    int y=5;
    printf("Valore di x: %d\n Valore di y: %d\n", x, y);
}
```

Valore di x: 3

Valore di y: 5

4

Esempi di printf



```
int q1=1;
int q2=4;
int q3=9;
int q4=16;
printf("I Quadrati di 1, 2, 3, 4: %d %d
%d %d %d \n",q1, q2, q3, q4);
```

Messaggio visualizzato:

I Quadrati di 1, 2, 3, 4: 1 4 9 16

5

Riepilogo: istruzione di ingresso (input)



Scanf: funzione che permette di prendere valori in ingresso dalla tastiera

Sintassi

`scanf(<stringa di controllo del formato>, <argomenti>)`

<stringa di controllo del formato>:

indica il formato in cui saranno inseriti i valori e il tipo di dato che verrà immesso dall'utente

<argomenti>:

variabili che conterranno i valori inseriti, preceduti dal carattere & (ampersand)

es. `scanf("%d", &z)`

6

Riepilogo: istruzione di assegnamento



```
x = 3 + 5 - (z * 5);
```

- calcola il valore dell'espressione a destra del segno = ed lo assegna alla variabile x

7

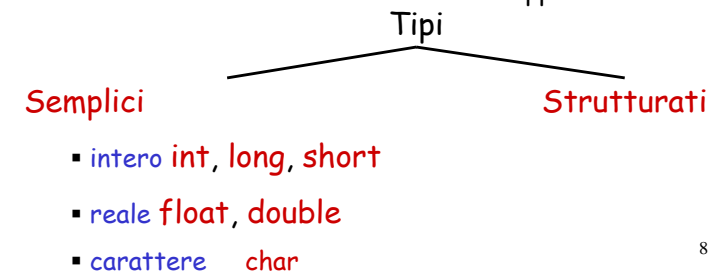
Tipi di dati



- **VALORI:** un insieme dei valori del tipo
- **OPERAZIONI:** per operare su tali valori

Utilità dei tipi di dato nelle dichiarazioni:

- allocazione di spazio
- operazioni più efficienti
- Esempio: $a + b$ si possono scegliere due operazioni diverse in linguaggio macchina a seconda se la addizione è tra interi oppure reali



8

Per ogni tipo vedremo



- Campo di variabilità
- Notazione per i valori costanti
- Operazioni
- Metodi di input/output

9

Tipo intero



Il tipo intero viene utilizzato per tutte le grandezze che possono essere rappresentate come numeri interi, come per es.: età, numero di figli, ecc.

Campo di variabilità (intervallo finito)

Tipo	Dimensione (byte)	Valore minimo	Valore massimo
short int	2	-32768	+32767
int	4	-2^{31}	$2^{31} - 1$
long int	4	-2^{31}	$2^{31} - 1$

10

Tipo intero



Notazione per i valori costanti

- Sequenza di cifre, preceduta dal segno - per i negativi.

```
int x;  
x = 356;  
x = -987;
```

Metodi di input/output

```
int x;
```

Output

```
printf("%d", x);
```

Input

```
scanf("%d", &x);
```

11

Esempi



```
#include <stdio.h>  
main()  
{  
  int x=9;  
  int y=6;  
  printf("x/y=%d\n", x/y);  
}
```

```
#include <stdio.h>  
main()  
{ short int x;  
  printf("Dammi uno short intero:");  
  scanf("%d", &x);  
  printf("%d", x);  
}
```

12

Operatori



- Gli **operatori** sono caratteri a cui è associata una operazione
- Gli **Operatori aritmetici** sono: + - * / % (modulo, ovvero resto della divisione intera)
- Consigli:
 - separare operatori con spazi migliora la lettura del programma!
 - scrivere codice leggibile utilizzando spazi!

```
a=k*(9.12+h)/y;
a=k*(9.12+h)/y;
```

Operazione	Operatore aritmetico	Espressione algebrica	Espressione C
Addizione	+	$f + 7$	$f + 7$
Sottrazione	-	$p - c$	$p - c$
Moltiplicazione	*	bm	$b * m$
Divisione	/	x / y	x / y
Modulo	%	$r \text{ mod } s$	$r \% s$

Esempi



```
int x=9;    x/y    |||>>>    1
int y=6;    x%y   |||>>>    3
```

Se $y=0 \rightarrow$ errore fatale, cioè che causa la terminazione immediata del programma, perché il computer non è in grado di eseguire una divisione per zero

Per i numeri negativi, la direzione di troncamento del / ed il segno del risultato di %, dipendono dalla macchina

Se $x\%y$ restituisce 0 \rightarrow x è multiplo di y o anche x è divisibile per y

Le parentesi nelle espressioni



Le parentesi sono utilizzate nelle espressioni C in maniera simile a quella utilizzata nelle espressioni algebriche.

Quindi le espressioni o parti di esse contenute all'interno di coppie di parentesi sono valutate per prime.

```
(a) x = z + 3 * (y + 2);
```

```
(b) x = x+2;
```

inizio

memoria

x	3
y	5
z	2

dopo (b)

memoria

x	3 23 25
y	5
z	2

(b)
Calcola valore dell'espressione
(23+2) = 25
Assegna tale valore alla variabile x

Precedenza degli operatori aritmetici



Il C valuta le espressioni aritmetiche in una sequenza precisa, determinata da *regole di priorità degli operatori*

che sono generalmente le stesse adottate in algebra,

ad esempio, le moltiplicazioni e le divisioni sono valutate prima delle somme e sottrazioni


Operatori	(Quando sono molti) Associatività
Parentesi () Valutate per prima.	dall'interno all'esterno
Operatore unario -	da destra a sinistra
Operatori binari * / %	da sinistra a destra
Operatori binari: + -	da sinistra a destra

+ alta


+ bassa

Associatività



$2*3*5$  $((2*3)*5)$
da sinistra a destra

$3*5+2$  17
* precedenza su +

$2*3/2*3$  * e / stessa precedenza:
sx verso dx: $((2*3)/2)*3$

$3*(5+2)$  21

$7+3-15+4*5$  15
 $((7+3)-15)+(4*5)$

17

Esercizio 1



- Scrivere un programma che richiede all'utente un numero che rappresenta un periodo di tempo espresso in minuti. Il programma converte tale periodo in ore e minuti e visualizza il risultato in ore e minuti

Esempio

Utente immette 134m \rightarrow 2 h, 14 m

Utente immette 45m \rightarrow 0 h, 45 m

Utente immette 180m \rightarrow 3 h, 0 m

18

Soluzione: esercizio 1



```
#include<stdio.h>

main()
{
    int numero, minuti, ore;

    printf("Dammi il tempo in minuti ");
    scanf("%d", &numero);
    ore = numero/60;
    minuti= numero%60;
    printf("%d h, %d m", ore, minuti);
}
```

Esercizio 2



Rovesciare un numero positivo di tre cifre

Esempio:

356 diventa 653

789 diventa 987

20



```
#include<stdio.h>

main()
{
  int n, unita, decine, centinaia;
  printf("Introduci un numero positivo di tre cifre\n");
  scanf("%d", &n);
  unita = n % 10;
  decine = (n/10) % 10;
  centinaia = n/100;
  printf("Numero rovesciato: %d%d%d",
        unita, decine, centinaia);
}
```

21



```
#include<stdio.h>
main()
{
  int n, unita, decine, centinaia;
  printf("Introduci un numero positivo di tre cifre\n");
  scanf("%d", &n);
  unita = n % 10;
  decine = (n/10) % 10;
  centinaia = n/100;
  printf("Rov: %d", 100*unita+10*decine+centinaia);
}
```

22



- Servono a incrementare (o a decrementare) di 1 una variabile.

++ aggiunge uno

-- sottrae uno

x++ equivale a **x=x+1**

x-- equivale a **x=x-1**

- Bisogna considerare l'esecuzione di due fasi:
 - Operazione (effettua l'incremento o decremento)
 - Valutazione per l'uso nella espressione in cui appare

- **Operatore postfisso:** es. **x++** o **x--**

L'espressione in cui appare la variabile utilizza il suo valore corrente, e solo in seguito questa variabile sarà incrementata (o decrementata)

- **Operatore prefisso:** es. **-x** o **++x**

prima viene incrementata (o decrementata) la variabile, poi il nuovo valore della variabile sarà utilizzato nell'espressione in cui appare

23



- La variabile viene comunque incrementata
- Attenzione alle istruzioni meno semplici!

x++; equivale a **x=x+1;**

++x; equivale a **x=x+1;**

y=x++; equivale a **y=x; x=x+1;**

y=++x; equivale a **x=x+1; y=x;**

Attenzione!

++(x+1) è un errore di sintassi

24



```
int n, m=0;
n=m++;
```

n vale 0
m vale 1

```
int n, m=0;
n=++m;
```

n vale 1
m vale 1

```
int n, m=0;
n=m=m+1;      equivale a(n=(m=m+1));
n vale 1
m vale 1
```



Il C fornisce gli operatori di assegnamento per abbreviare le istruzioni che utilizzano espressioni.

Ad esempio :

```
numero_studenti = numero_studenti + 10;
```

può essere scritto in maniera più compatta come:

```
numero_studenti += 10;
```

+= è l'operatore di assegnamento addizione

L'operatore += somma il valore dell'espressione a destra con il valore della variabile a sinistra e poi memorizza il risultato nella variabile a sinistra.



L'operatore di assegnamento si può usare con tutti gli operatori aritmetici visti (e poi con altri che vedremo)

+= -= *= /= %=

Esempi di espressioni con operatori di assegnamento

```
d -= 4      (d = d - 4)
```

```
e *= 5      (e = e * 5)
```

```
f /= 3      (f = f / 3)
```

```
g %= 9      (g = g % 9)
```



La priorità e l'associatività degli operatori introdotti sino a questo punto.

Notare gli operatori che associano da destra!

Operatori	Associatività
() ++ (postfisso) -- (postfisso)	Da sinistra
+ - (unari) ++(prefisso) -- (prefisso)	Da destra
* / %	Da sinistra
+ -	Da sinistra
= += -= *= /= %= ...	Da destra

Alcuni esempi di espressioni



```
int a = 1, b = 2, c = 3, d = 4;
```

```
a * b / c
```

Operatori * e / con la stessa priorità, **associatività** da sinistra.

```
(a * b) / c
```

Operatori	Associatività
() ++ (postfisso) -- (postfisso)	Da sinistra
+ - ++(prefisso) -- (prefisso)	Da destra
* / %	Da sinistra
+ -	Da sinistra
= += -= *= /= %= ...	Da destra

Attenzione! Conversione di tipo automatica in intero

Alcuni esempi di espressioni



```
int a = 1, b = 2, c = 3, d = 4;
```

```
a * b % c + 1
```

Operatori * e % con la stessa priorità, associatività da sinistra poi operatore +.

```
((a * b) % c) + 1
```

Operatori	Associatività
() ++ (postfisso) -- (postfisso)	Da sinistra
+ - ++(prefisso) -- (prefisso)	Da destra
* / %	Da sinistra
+ -	Da sinistra
= += -= *= /= %= ...	Da destra

Alcuni esempi di espressioni



```
int a = 1, b = 2, c = 3, d = 4;
```

```
++ a * b - c --
```

Operatore -- postfisso, operatore ++ prefisso, operatore * e -

```
((++ a) * b) - (c --)
```

Operatori	Associatività
() ++ (postfisso) -- (postfisso)	Da sinistra
+ - ++(prefisso) -- (prefisso)	Da destra
* / %	Da sinistra
+ -	Da sinistra
= += -= *= /= %= ...	Da destra

Attenzione! L'operatore postfisso effettua la operazione dopo la valutazione

Alcuni esempi di espressioni



```
int a = 1, b = 2, c = 3, d = 4, e = 5;
```

```
5 + c * -- d / e
```

Operatore -- prefisso, operatori * e / (assoc. da sinistra), operatore +

```
5 + ((c * (-- d)) / e)
```

Operatori	Associatività
() ++ (postfisso) -- (postfisso)	Da sinistra
+ - ++(prefisso) -- (prefisso)	Da destra
* / %	Da sinistra
+ -	Da sinistra
= += -= *= /= %= ...	Da destra

Alcuni esempi di espressioni



```
int a = 1, b = 2, c = 3, d = 4, e = 5 ;
```

30 / -++ e - + 29 % c
Operatore ++ prefisso,
operatori - e + unario (assoc. da destra)

30 / (- (++ e)) - (+ 29) % c
Operatori / e % (assoc. da sinistra)
Operatore -

(30 / (- (++ e))) - ((+ 29) % c)

Operatori	Associatività
() ++ (postfisso) -- (postfisso)	Da sinistra
+ - ++ (prefisso) -- (prefisso)	Da destra
* / %	Da sinistra
+ -	Da sinistra
= += -= *= /= %= ...	Da destra

33

Esercizi



Valutare le seguenti espressioni, tenendo presente che:

```
int a = 1, b = 2, c = 3, d = 4, e = 5 ;
```

1 5 + -- a - e / b ++

2 - a --- b --- c

3 a / b % e ++

34

Tipo reale



- I numeri reali vengono usati per rappresentare prezzi, pesi, misure, per calcoli matematici, ecc.
- Sono espressi con una virgola decimale (detti numeri in virgola mobile)

Campo di variabilità (Intervallo finito)

Tipo	Dimensione (byte)	Valore minimo	Valore massimo
float precisione singola	4	-3.2·10 ^{±38}	+3.2·10 ^{±38}
double precisione doppia	8	-1.7·10 ^{±308}	+1.7·10 ^{±308}

35

Tipo reale



Operazioni: somma, differenza unaria e binaria, prodotto, divisione reale

Altre operazioni come esponenziali, logaritmi, funzioni trigonometriche sono possibili utilizzando le funzioni della libreria standard math.h

```
#include <math.h>
/* x e y di tipo double e restituiscono un double */
pow(x, y)          xy
sin(x)             seno di x, con x espresso in radianti
cos(x)             coseno di x, con x espresso in radianti
exp(x)             ex
log(x)             logaritmo naturale di x
sqrt(x)            radice quadrata x, x>=0
log10(x)           logaritmo in base 10 di x
.....
```

36

Tipo reale



Notazione per i valori costanti

Esistono due modi di scrivere i numeri reali

1. parte intera punto parte decimale 4.34
2. parte intera e o E esponente con segno
-3E3 rappresenta $-3 \cdot 10^3$ cioè -3000
5e-2 rappresenta $5 \cdot 10^{-2}$ cioè 0.05

Metodi di input/output

```
float x; (o double x);
```

Output

```
printf("%f", x);
```

Input

```
scanf ("%f", &x);
```

```
printf("%.2f", x);
```

Il .2 è la precisione con cui il valore sarà visualizzato.

Il .2 dice che il valore verrà visualizzato con due cifre a destra della virgola.

Con %f per default si usano 6 cifre

```
printf("%.2f\n", 3.446);
```

Visualizza 3,45

Esercizio 1



- Scrivere un programma che calcola l'area di un triangolo di base b ed altezza h , immessi dall'utente

Soluzione: esercizio 1



```
#include<stdio.h>
main()
{
    float base, altezza, area;
    printf("Dammi la base: ");
    scanf("%f", &base);
    printf("Dammi l'altezza: ");
    scanf("%f", &altezza);
    area = (base * altezza)/2;
    printf("L'area: %f\n", area);
}
```

Esercizio 2



- Scrivere un programma che effettua la conversione da EURO a LIRE ITALIANE

Esempio

Se immetto 100 EURO

100 Euro = 193627 LIRE

Soluzione: esercizio 2



```
#include <stdio.h>
main()
{
    float euro, lira;
    printf("Dammi il numero in EURO: ");
    scanf("%f", &euro);
    lira = euro * 1936.27;
    printf("%f EURO = %f LIRE", euro, lira);
}
```

Tipo carattere



- I computer elaborano informazioni costituite non solo da numeri, ma anche da testi, che a loro volta sono composti da *caratteri*
- Codice ASCII : rappresenta 128 simboli diversi (codice di 7 bit):
- lettere dell'alfabeto, cifre, segni di punteggiatura e altri simboli

Campo di variabilità (Intervallo finito)

Tipo	Dimensione (byte)
char	1

Codice ASCII



Caratter	Decimale	Binario
e		
{	123	1111011
a	97	1100001
A	65	1000001
B	66	1000010
,	59	0111011
3	51	0110011
&	38	0100110

- ad ogni carattere corrisponde una rappresentazione numerica univoca
- i caratteri sono totalmente ordinati
- 0 ... 9
valori nell'intervallo 48 ...57
- A ... Z
valori nell'intervallo 65 ...90
- a ... z
valori nell'intervallo 97 ... 122

'A' è maggiore di ',' che è maggiore di '&'

Si ha che: a<b<c<.....<z

A<B<C<.....<Z

0<1<2<.....<9

Notazioni per valori costanti



Per assegnare un valore costante ad una variabile char lo si deve racchiudere tra apici, es.

x = 'A';

y = ',';

z = '&';

per quelli non stampabili si usa \ (*sequenza di escape*)

\n a capo

\\ backslash

\t tabulazione

\' carattere apice

\" doppio apice

Le sequenze di escape vanno racchiuse tra apici come i simboli dei caratteri



Operazioni

restituire il carattere che segue/precede;

- operazioni di uguaglianza/disuguaglianza;
- chiedersi se un carattere è maggiore/minore di un altro

Metodi di input/output

```
char x;
```

output

```
printf("%c", x);
```

input

```
scanf ("%c", &x);
```

45



- In C il tipo `char` è un tipo intero su 1 byte utilizzato principalmente per rappresentare caratteri

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

```
main()
```

```
{
```

```
    char x=65;
```

```
    printf("%c", x);    /* stampa A */
```

```
    printf("%d", x);    /* stampa 65 */
```

```
    x='A';
```

```
    x=x+1;
```

```
    printf("%c", x);    /* stampa B */
```

```
}
```

46



- Scrivere un programma che legge prima tre caratteri e poi li stampa in ordine inverso.

47



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

```
main()
```

```
{
```

```
    char x, y, z;
```

```
    printf("Digita tre caratteri: ");
```

```
    scanf("%c%c%c", &x, &y, &z);
```

```
    printf("Hai digitato (ordine  
                inverso): ");
```

```
    printf("%c%c%c", z, y, x);
```

```
}
```

48

Esercizio



- Scrivere un programma che legge prima un carattere e poi visualizza a video il carattere che lo segue e quello che lo precede

Esempio:

Se legge

D

deve visualizzare a video:

E C

49

Soluzione



```
#include <stdio.h>
main()
{
    char x;
    printf("Digita un carattere: ");
    scanf("%c", &x);
    printf("Il succ = %c\n", x+1);
    printf("Il prec = %c\n", x-1);
}
```

50

Conversioni



Il risultato di un'operazione dipende dal tipo di dato che è coinvolto nell'operazione

```
int x=5;
int y=2;
... x/y ... vale 2
```

```
float x=5;
float y=2;
... x/y ... vale 2.5
```

- a seconda del tipo di dato coinvolto nelle operazioni, le operazioni sono svolte in maniera diversa
- Dividere due interi produce una divisione tra interi in cui ogni parte frazionaria del calcolo è persa (troncata).
- Dividere due reali produce una divisione tra reale e quindi con il risultato reale

51

Le conversioni di tipo



- I valori di una variabile possono essere convertiti da un tipo all'altro
 - implicitamente (dal compilatore)
 - esplicitamente (dal programmatore)

52

Conversioni implicite



- Se due operandi di un'operazione sono tra loro diversi, l'operando di tipo più piccolo viene convertito nel **tipo più grande**, senza perdita di informazioni
- tale operazione è fatta dal compilatore e viene chiamata **promozione**

```
int i; float f;
```

```
...
```

```
f=f*i;      i convertito in float
```

- Nell'assegnamento, il C converte il valore del lato destro nel tipo del **lato sinistro**, eventualmente **perdendo informazioni** quando si passa da un tipo più grande ad un tipo più piccolo

```
int i;
```

```
float f=3.6;
```

```
i=f;
```

i vale 3

53

Conversioni esplicite



- Il programmatore può richiedere esplicitamente la conversione di un valore da un tipo ad un altro utilizzando l'**operatore di cast** o di **conversione**:

(nome_nuovo_tipo) espressione

Ad esempio se voglio una divisione reale (con la virgola) tra interi e quindi ottenere un risultato reale includo l'operatore di cast (**float**) che creerà una variabile temporanea che include il valore in virgola mobile della variabile a cui è applicato:

```
int x=5;
```

```
int y=2;
```

```
float f;
```

```
f=(float)x/(float)y;
```

f vale 2.5

54

Nota: cosa stampa??



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

```
main()
```

```
{
```

```
int x=5;
```

```
int y=2;
```

```
float f;
```

```
f=(float) (x/y);
```

```
printf("%f", f);
```

```
}
```

2.0

55

Nota



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

```
#include <math.h>
```

```
main()
```

```
{int x=9;
```

```
printf("%d\n", sqrt(x));
```

```
}
```

stampa

0

sqrt si aspetta come argomento un double e, se si trova ad operare su un qualcosa di diverso, produce un risultato privo di significato

56



```
#include <stdio.h>
#include <math.h>
main()
{int x=9;
  printf("%f\n", sqrt((double) x));
  // stampa 3.0
}
```

Valore corretto, ma x
rimane inalterato

