Input / Output

cin >> gestisce input (vedi esempi)

cout << gestisce output (vedi esempi)

in **cout** le seguenti istruzioni, scritte tra virgolette, danno i seguenti risultati:

\n porta il cursore all'inizio della riga successiva

\t porta il cursore al prossimo fermo di tabulazione (ogni fermo di tabulazione è fissato ad 8 caratteri) tabulazione orizzontale

\v tabulazione verticale

\b back space

\\ stampa la barra rovesciata

\? stampa il punto interrogativo

\' stampa un apice

\" stampa le virgolette

Di default **cout** stampa 6 (7 a seconda dei compilatori) cifre decimali**; per modificarle abbiamo 2 vie:**

- utilizzare **setprecision**(numero_cifre):

  cout << setprecision(12) << x; (x viene stampata con 12 cifre)

  il comando è persistente, imposta l'output fino a nuova definizione
  è necessario aggiungere in testa al file
  # include<iomanip> libreria che permette la manipolazione dell'output

- Scrivere prima del primo uso di cout
  cout.precision(numero_cifre)

  anche questo è persistente ma non richiede di includere iomanip.

Input e output su file

Necessario : #include <fstream.h>

Output

ofstream prt("myfile") // se non esiste già, crea il file myfile nella current directory

- dopo questa istruzione si utilizza il comando ***prt*** al posto di cout e l'output avviene sul file myfile al posto del monitor
- l'istruzione ofstream posiziona l'inizio della scrittura in testa al file: se il file non è vuoto sovrascrivo i dati e perdo quelli vecchi
- i manipolatori dell'output come setprecision si utilizzano ancora con la stessa sintassi; attenzione: cout.precision() diventa prt.precision()

Input

ifstream read("myfile") // legge dal file ***myfile*** nella current directory;

- dopo questa istruzione si utilizza il comando ***read*** al posto di cin
- l'istruzione ifstream posiziona l'inizio della lettura in testa al file
- i valori nel file ***myfile*** devono essere separati da spazi o da return

N.B. Le due parole chiave ***prt*** e ***read*** sono arbitrarie, a discrezione del programmatore che le può scegliere a piacere

Funzioni e Parametri di default

Molte funzioni includono argomenti ai quali si vorrebbe assegnare, se non diversamente specificato, un valore di default.
A questi argomenti, che possono essere omessi nella chiamata alla funzione, può essere assegnato un valore di default nel prototipo della funzione.

Esempio

Definisco la funzione

```
int f(int a, int b){
return a+b;
}
```

Prima del main:

```
int f(int a, int b=0); // prototipo di f
```

Se nel main scrivo:

```
a = f(3,2);  //  a = 5
```

Ma se scrivo

```
a = f(3);  //  a=3 perché b se non specificato vale 0
```

N.B. L'indicazione del valore di default va messa **SOLO** nel **prototipo**

N.B. Gli argomenti di default **DEVONO** essere gli ultimi sulla destra e vanno omessi ordinatamente da quello più a destra a scalare, senza lasciare "buchi".

Puntatori a Funzioni

Scopo:

- vogliamo che una funzione accetti un'altra funzione come argomento in input
- voglio definire un puntatore ad un particolare tipo di funzioni e farlo puntare ad una funzione di quel tipo

Esempio

Ho una funzione che calcola la somma dei primi M termini di una successione. Voglio che riceva in input la funzione che definisce il termine n-esimo della successione.

(vedi esempio)

Funzioni e Variabili *Static*

Se in una funzione si dichiara una variabile "static" questa, dopo l'uscita del programma dalla funzione, non verrà cancellata ma resterà a disposizione per la chiamata successiva della funzione.

Esempio

```
int f(int a, int b){
static int n;
n++;
cout  << n << endl;
return a+b;
}
```

se non inizializzata diversamente verrà inizializzata a zero