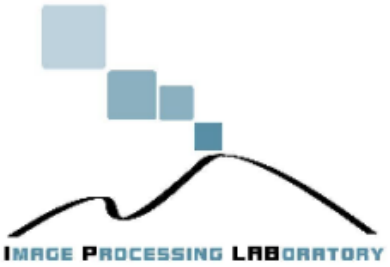
Gli stream

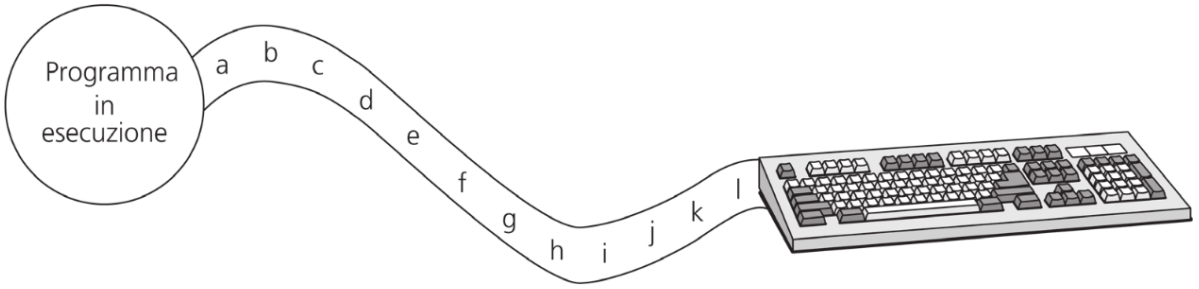
Alessandro Ortis
Image Processing Lab - iplab.dmi.unict.it

alessandro.ortis@unict.it
www.dmi.unict.it/ortis/



Flusso (stream)

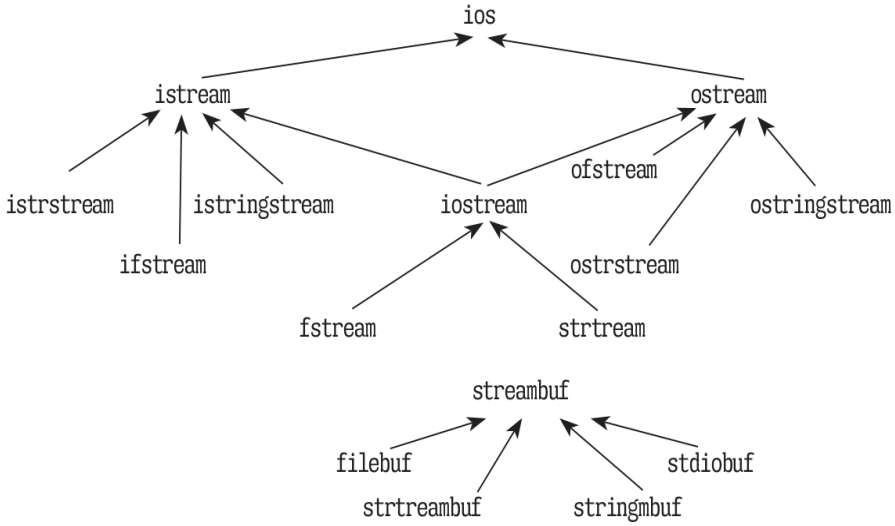
- astrazione che rappresenta un flusso di dati che scorre tra un programma produttore ed un programma consumatore
- "standard input": corrente di bytes che fluisce (normalmente) dalla tastiera verso un programma in esecuzione sulla CPU
- "standard output" come un flusso che scorre dalla CPU verso un dispositivo di uscita
- "estrarre da un flusso": ricezione dati da un dispositivo di input
- "inserire in un flusso": trasmissione di dati ad un dispositivo di output.



Libreria <iostream>

Distribuita su quattro header files:

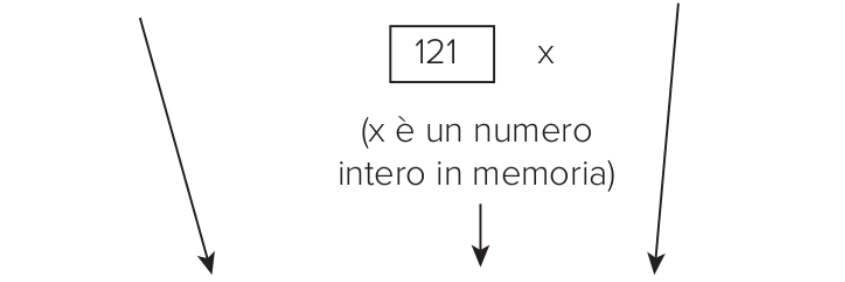
- <iostream> classi istream, ostream ed iostream per le operazioni di I/O con i flussi di input ed output standard, nonché gli oggetti cout, cin, cerr e clog che si utilizzano nella maggior parte dei programmi C++
- <fstream> classi ifstream, ofstream ed fstream per operazioni di I/O su files (memoria di massa).
- <sstream> classi istringstream ed ostringstream.
- <strstream> classi istrstream, ostrstream e strstream per formattare dati con buffers di caratteri.



Conversione di dati a/da flussi di caratteri

Conversione in uscita (da rappresentazione interna a caratteri)

```
cout << "Introduca " << setw(3) << x << " valori interi ";
```

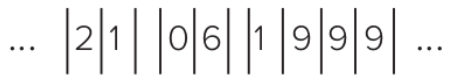


`cout` (flusso di caratteri in uscita verso lo schermo)

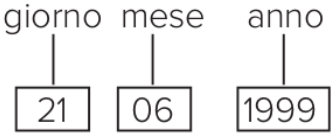
Conversione in ingresso

```
cin >> giorno >> mese >> anno;
```

`cin` (flusso di caratteri letti dalla tastiera)



Rappresentazione interna degli interi letti



Oggetti di <iostream>

Oggetto di flusso	Funzione
cin	Oggetto della classe istream collegato all'input standard
cout	Oggetto della classe ostream collegato all'output standard
cerr	Oggetto della classe ostream collegato all'errore standard, per output senza buffer
clog	Oggetto della classe ostream collegato all'errore standard, per output tramite buffer

istream

- la classe istream permette di definire un flusso d'input e contiene metodi per accettare dati in modo formattato e non formattato

- per poter effettuare operazioni di input ad alto livello l'estrattore >> è sovraccaricato per tutti i tipi di dato fondamentali del C++: `char`, `unsigned char`, `signed char`, `short`, `unsigned short`, `int`, `unsigned int`, `long`, `unsigned long`, `float`, `double`, `long double`, stringhe e puntatori.

Indicatori di stato

Chiamata a funzione	Restituisce *true* se e solo se
`cin.good()`	è tutto corretto in `cin`
`cin.bad()`	c'è qualcosa di sbagliato in `cin`
`cin.fail()`	non s'è potuta completare l'ultima operazione

ios_base::iostate flags			basic_ios accessors					
eofbit	failbit	badbit	good()	fail()	bad()	eof()	operator bool	operator!
false	false	false	true	false	false	false	true	false
false	false	true	false	true	true	false	false	true
false	true	false	false	true	false	false	false	true
false	true	true	false	true	true	false	false	true
true	false	false	false	false	false	true	true	false
true	false	true	false	true	true	true	false	true
true	true	false	false	true	false	true	false	true
true	true	true	false	true	true	true	false	true

Conversione in output di diversi tipi

Tipo	Tipo di conversione di output
`char`	I caratteri stampabili si visualizzano con la larghezza di una colonna. I caratteri di controllo, come nuova riga, tabulazione ecc., possono produrre più caratteri di output
`int`	Qualunque tipo intero (`int`, `short` o `long`) si visualizza come numero decimale con larghezza sufficiente per contenere il numero e il segno meno se l'intero fosse negativo
Stringa	La larghezza su schermo è uguale alla lunghezza della stringa
`float`	I numeri reali in virgola mobile si visualizzano con la precisione di sei cifre decimali. Gli zeri non significativi non vengono visualizzati. Se il numero è molto grande o molto piccolo, si visualizza il numero con un esponente di due cifre (o tre cifre se il tipo è `double`) dopo la lettera "`e`". La larghezza è sempre sufficiente per mantenere un segno meno e/o un esponente

Formattazione dell'output

Manipolatore	Azione
`dec`	utilizza conversione decimale (*per default*)
`hex`	utilizza conversione esadecimale
`oct`	utilizza conversione ottale
`ws`	estrae caratteri spazi in bianco
`endl`	aggiunge il carattere *newline* al flusso di output (`'\n'`)
`ends`	aggiunge il carattere terminale nullo al flusso di output (`'\0'`)
`flush`	svuota il buffer di output inviando immediatamente i dati nel flusso di output
`setbase(n)`	stabilisce la base di conversione a *n* (0, 8, 10 oppure 16). 0 significa decimale per default
`setprecision(n)`	stabilisce la precisione di virgola mobile a *n*
`setw(n)`	stabilisce la larghezza del campo a *n*
`setfill(c)`	stabilisce il carattere di riempimento a *c*
`setiosflags(f)`	setta i bit di formato specificati dall'argomento *f* di tipo long
`resetiosflags(f)`	azzera i bit di formato specificati dall'argomento *f* di tipo long

I/O da file

- un file è semplicemente un flusso esterno: una sequenza di bytes in memoria di massa: se il file viene aperto per scriverci è un flusso di output, se viene aperto per leggervi dentro è un flusso di input.
- tre classi:
  `ifstream`, per aprire un file di testo in lettura
  `ofstream` per aprire un file in scrittura
  `fstream` per aprire un file indistintamente
- per aprire il file in lettura bisogna dichiarare un oggetto flusso ed associarlo ad un file:
  ```
  ifstream fin("demo.txt");
  ```
  l'oggetto si chiama `fin` ed è associato ad un file il cui nome è "`demo.txt`"
- per aprire il file in scrittura non basta dichiarare un oggetto flusso ed associarlo ad un file ma bisogna specificare anche la modalità d'apertura in un secondo parametro:
  ```
  ofstream fout("demo.txt", ios::out);
  ```
  perché un flusso `ofstream` in scrittura si può aprire in due modi:
  `output(ios::out)` per sovrascrivere i file ed `append (ios::app)` per continuare a scrivere nel file dopo la fine
- per default, un file ofstream si apre in modo output:
  ```
  ofstream fout ("demo");
  ```

Valori dell’argomento modo di open

Le modalità di apertura sono comunque varie

Argomento	Modo
ios::in	Modo input
ios::app	Modo append
ios::out	Modo output
ios::ate	Aprire e cercare la fine del file
ios::nocreate	Genera un errore se non esiste il file
ios::trunc	Tronca il file a 0 se esiste già
ios::noreplace	Genera un errore se il file esiste già
ios::binary	Il file si apre in modo binario

Ridirezione degli stream standard

Un modo comune per ridirezionare uno stream di default (es. stdin, stdout e stderr) è l'utilizzo della funzione reopen(). Questo metodo verifica se uno stream è già aperto, ed eventualmente lo riapre.

```
// Ridirezione dello stdout
#include <stdio.h>
#include <iostream>
int main ()
{
  freopen ("myfile.txt","w",stdout);
  printf ("Questa frase e' ridirezionata su un file.\n");
  cout << ''anche questa'' << endl;
  fclose (stdout);
  return 0;
}
```

I/O binario

- files che rappresentano immagini, suoni o filmati, non sono file di testo, ovvero i bytes non sono pensati per rappresentare testo

- Per scrivere e leggere dati in formato binario da/a un file si utilizzano le funzioni:
  - `put()`
  - `get()`
  - `read()`
  - `write()`