



Capitolo 10

Flussi e file: libreria standard di Input/Output

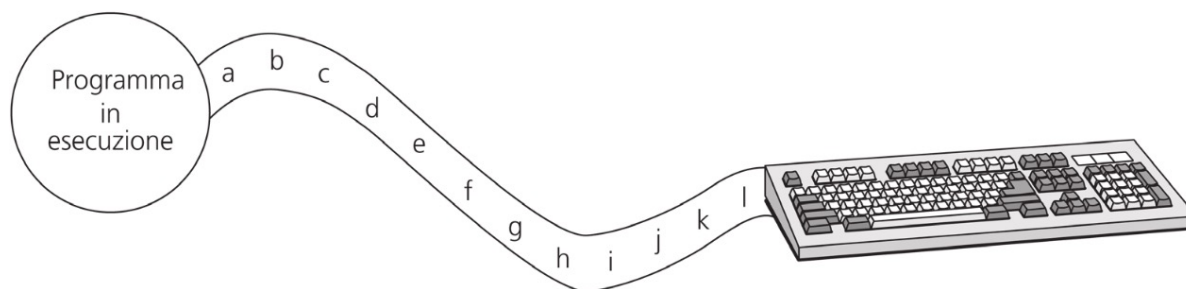
pag. 255-286

Presenta: Prof. Misael Mongiovì



Flusso (stream)

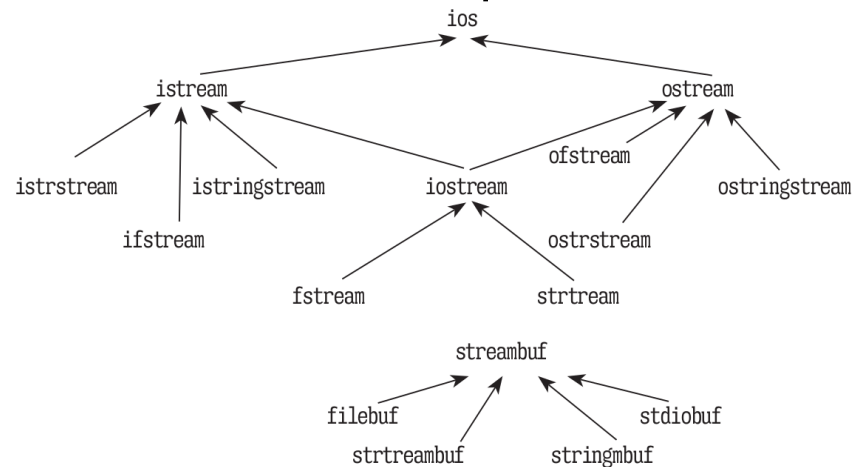
- astrazione che rappresenta un flusso di dati che scorre tra un programma produttore ed un programma consumatore
- "standard input": corrente di bytes che fluisce (normalmente) dalla tastiera verso un programma in esecuzione sulla CPU
- "standard output" come un flusso che scorre dalla CPU verso un 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>`

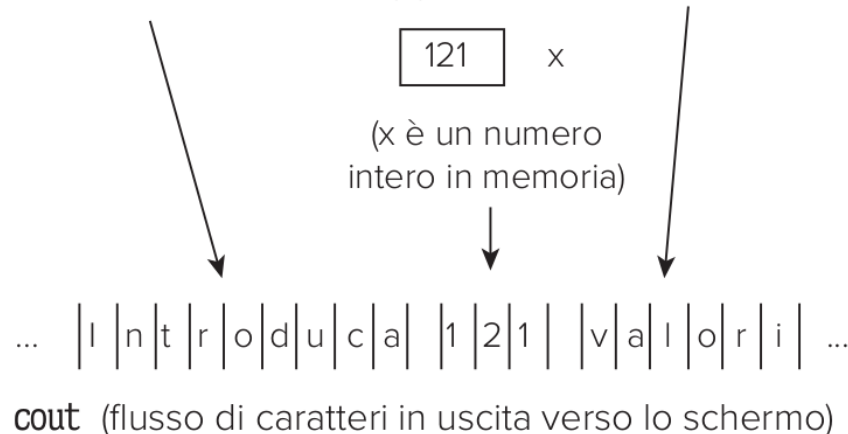
- *al momento* distribuita su quattro header files:
 - `<iostream>` classi `istream`, `ostream` ed `iostream` per le operazioni di I/O con i flussi di input ed output standard, nonché gli oggetti `cout`, `cin`, `cerr` e `clog` che si utilizzano nella maggior parte dei programmi C++ (Tabella 10.1)
 - `<fstream>` classi `ifstream`, `ofstream` ed `fstream` per operazioni di I/O su files (memoria di massa).
 - `<sstream>` classi `istringstream` ed `ostringstream`.
 - `<sstream>` 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 ";
```



Conversione in ingresso

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

`cin` (flusso di caratteri letti dalla tastiera)





oggetti di `<iostream>`

Oggetto di flusso	Funzione
<code>cin</code>	Oggetto della classe <code>istream</code> collegato all'input standard
<code>cout</code>	Oggetto della classe <code>ostream</code> collegato all'output standard
<code>cerr</code>	Oggetto della classe <code>ostream</code> collegato all'errore standard, per output senza buffer
<code>clog</code>	Oggetto della classe <code>ostream</code> 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 <i>true</i> se e solo se
<code>cin.good()</code>	è tutto corretto in <code>cin</code>
<code>cin.bad()</code>	c'è qualcosa di sbagliato in <code>cin</code>
<code>cin.fail()</code>	non s'è potuta completare l'ultima operazione

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
<i>Stringa</i>	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 (<i>per default</i>)
hex	utilizza conversione esadecimale
oct	utilizza conversione ottale
ws	estrae caratteri spazi in bianco
endl	aggiunge il carattere <i>newline</i> al flusso di output (' <code>\n</code> ')
ends	aggiunge il carattere terminale nullo al flusso di output (' <code>\0</code> ')
flush	svuota il buffer di output inviando immediatamente i dati nel flusso di output
setbase(n)	stabilisce la base di conversione a <i>n</i> (0, 8, 10 oppure 16). 0 significa decimale per default
setprecision(n)	stabilisce la precisione di virgola mobile a <i>n</i>
setw(n)	stabilisce la larghezza del campo a <i>n</i>
setfill(c)	stabilisce il carattere di riempimento a <i>c</i>
setiosflags(f)	setta i bit di formato specificati dall'argomento <i>f</i> di tipo long
resetiosflags(f)	azzerà i bit di formato specificati dall'argomento <i>f</i> 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");`
l'oggetto si chiama `fin` ed è associato ad un file il cui nome è "demo"
- 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", 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
<code>ios::in</code>	Modo input
<code>ios::app</code>	Modo append
<code>ios::out</code>	Modo output
<code>ios::ate</code>	Aprire e cercare la fine del file
<code>ios::nocreate</code>	Genera un errore se non esiste il file
<code>ios::trunc</code>	Tronca il file a 0 se esiste già
<code>ios::noreplace</code>	Genera un errore se il file esiste già
<code>ios::binary</code>	Il file si apre in modo binario



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()`



accesso diretto al file

- si può accedere direttamente ai singoli bytes dei flussi files tramite le funzioni:
 - `seekg()`
 - `tellg()`

con la seguente sintassi:

```
istream& seekg (streampos pos);
```

```
istream& seekg (streamoff scostamento, seekdir ind);
```

```
streampos tellg();
```