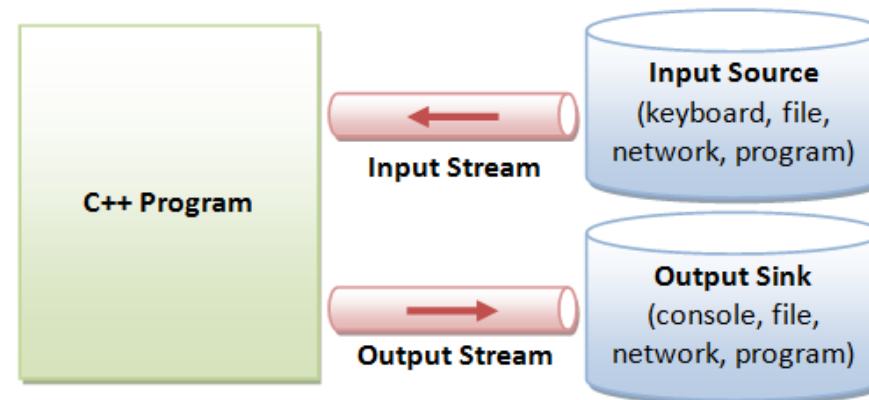




**UNIVERSITÀ
DI PARMA**

Informatica e Laboratorio di Programmazione
C++ stream
Alberto Ferrari

- l'*input/output* del C++ è basato sugli *stream*
- gli stream sono *sequenze di byte* che rappresentano il *flusso* in entrata o in uscita di un programma
- gli stream fungono da *intermediari* fra i programmi e le periferiche di I/O liberando il programmatore dalla necessità di gestire direttamente le periferiche



Internal Data Formats:

- Text: `char`, `wchar_t`
- `int`, `float`, `double`, etc.

External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

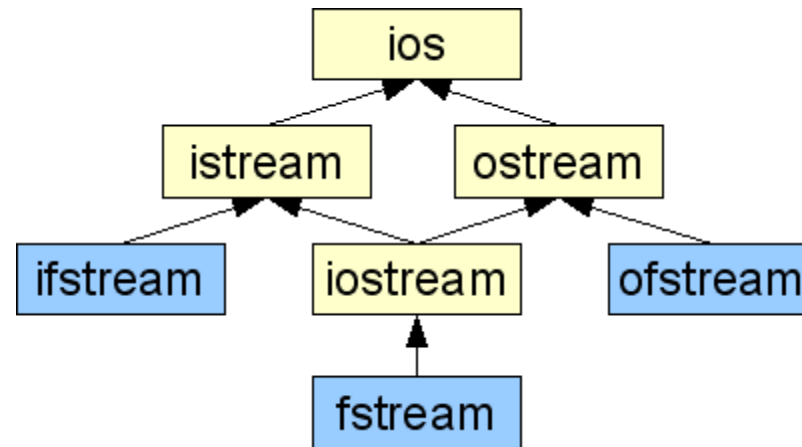
- nelle operazioni di input i dati vanno da una *sorgente di input* verso il *programma*
 - una sorgente di input può essere
 - la *tastiera* (*console input standard*)
 - un *file*
 - una *risorsa di rete*
 - l'*output* di un altro *programma*



- nelle operazioni di output il flusso di dati che ha come *sorgente* il *programma* può essere diretto verso:
 - il *video* (*console output standard*)
 - un *file*
 - la *rete*
 - un altro *programma*



- << e >> sono **operatori** di flusso che operano sugli **stream**
- << **estrazione** da uno stream
- >> **inserimento** in uno stream
- **cin, cout** (e **cerr**) sono oggetti che rappresentano stream di input, output ed error standard
 - console **input**
 - console **output**
 - console **error**



I/O file

C++ - file

- insieme di dati memorizzati su un supporto di memoria *non volatile*
- *globale* ai programmi
 - un file può essere scritto da un programma e letto da un programma diverso scritto anche in un altro linguaggio di programmazione
- la gestione concreta dei file è demandata al *file system* del sistema operativo
- il file system fornisce all'utente una *versione astratta* dell'organizzazione dei file
 - esistono diversi tipi di file system
 - tutti basati sul concetto di struttura gerarchica di cartelle (directory)
 - le cartelle vengono viste come file speciali

- **apertura**
 - necessaria per le successive operazioni di lettura e scrittura
 - controllo dei diritti di accesso e meccanismi di gestione della concorrenza
 - viene creato un buffer in memoria per la gestione dei dati in transito tra il programma e il file
- **chiusura**
 - non verranno più effettuate operazioni di lettura e scrittura
 - rilascio del buffer
 - operazione necessaria per la gestione della concorrenza
- **lettura**
 - dati trasferiti dal file nel buffer
 - gestione dei dati da parte del programma
- **scrittura**
 - dati memorizzati nel buffer (temporaneamente)
 - file system gestisce la scrittura fisica sul file

- ***file sequenziali***
 - accesso ai dati nello stesso ***ordine*** in cui sono stati inseriti
- ***ofstream (output file stream)***
 - ***output*** su memoria di massa
 - file con accesso in ***sola scrittura***
- ***ifstream (input file stream)***
 - ***input*** da memoria di massa
 - file con accesso in ***sola lettura***
- ***fstream (file stream)***
 - ***input e output*** su memoria di massa
 - file con accesso in ***lettura e/o in scrittura***

```
#include <iostream>
#include <fstream>

using namespace std;

int main() {
    ofstream f_out;           // output stream
    f_out.open("myfile.txt"); // open
    f_out << "first line" << endl; // write
    f_out << "second line" << endl;
    f_out.close();           // close
    return 0;
}
```

- myFile (oggetto *ofstream*) è il *nome logico* del file
 - nome che lo identifica all'interno del programma
- *open* (*apertura* del file e collegamento fra il nome logico e il nome fisico)
 - per ofstream il file viene in ogni caso *creato* e *riscritto* anche se già presente
- myfile.txt è il *nome fisico* (nome del file su disco (*pathname*))
- << inserimento in stream (*scrittura*)
- *close* (chiusura)
 - viene eliminato il buffer ed effettuata la scrittura fisica

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    string s;
    ifstream f_in;
    f_in.open("divina.txt");
    if (f_in.fail()) {
        cout << "file error" << endl;
        return 1;
    }
    f_in >> s;
    while(!f_in.eof()) {
        cout << s << endl;
        f_in >> s;
    }
    f_in.close();
    return 0;
}
```

- *ifstream* (input file stream)
- *open* apertura e associazione **nome logico - nome fisico**
- il file deve esistere (non viene creato) **fail** verifica se l'apertura ha avuto successo
- >> estrazione (*stringa*) dallo stream
 - il separatore può essere **spazio** o **endline**
- l'operazione di lettura può rivelare che il file è terminato
- *eof* (**end of file**) vale true se si è raggiunta la **fine del file**
- *close* (chiusura file)

```
#include <iostream>
#include <fstream>

using namespace std;

int main() {
    ifstream f_in{"divina.txt"};    //declare and open
    if (f_in.fail()) {
        cout << "file error" << endl;
        return 1;
    }
    // for each line in f_in...
    for (string line; getline(f_in, line);) {
        cout << line << endl;
    }
    f_in.close();
    return 0;
}
```

```
#include <iostream>
#include <fstream>

using namespace std;

int main() {
    ifstream f_in{"divina.txt"};    //declare and open
    if (f_in.fail()) {
        cout << "file error" << endl;
        return 1;
    }
    string whole_text;
    getline(f_in, whole_text, '\0'); // read whole file
    cout << whole_text;
    f_in.close();
    return 0;
}
```

```
int main() {
    fstream f1,f2;
    int r; char p,c;
    f1.open("temp.txt", ios::out);
    f1 << 'k' << 'e' << 1 << endl;
    f1 << 'Q' << 'd' << 8 << endl;
    f1.close();
    f2.open("temp.txt", ios::in);
    f2 >> p;
    while(!f2.eof()) {
        cout << p << "(";
        f2 >> c;
        cout << c << ",";
        f2 >> r;
        cout << r << ")" << endl;
        f2 >> p;
    }
    f2.close();
}
```

- *fstream* (file stream)
- *open*
 - *ios::out* (output)
 - il file viene creato
 - *ios::in* (input)
 - controllare fail
 - *ios::app* (append)
 - il file deve esistere
 - scritture in aggiunta al file precedente

C++
string stream

- si può gestire una *stringa* come uno *stream*
- `istringstream`, `ostringstream` in libreria `<sstream>`

```
#include <iostream>
#include <sstream>

using namespace std;

int main() {
    /* Split a text into a sequence of strings */
    string text = "one:two::three";
    istringstream sstr{text}; // a stream view on a string
    for (string item; getline(sstr, item, ':');) {
        cout << "- " << item << endl;
    }
}
```