Laurea triennale in Informatica

modulo **(CFU 6)** *di*

Programmazione II e Lab.

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Isola C4
stanza: n. 423 – IV piano Lato Nord
tel.: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

File e stream in C++

Il C++ supporta due File System completi: uno ereditato dal C e l'altro (proprio del C++) orientato agli oggetti.

File e stream

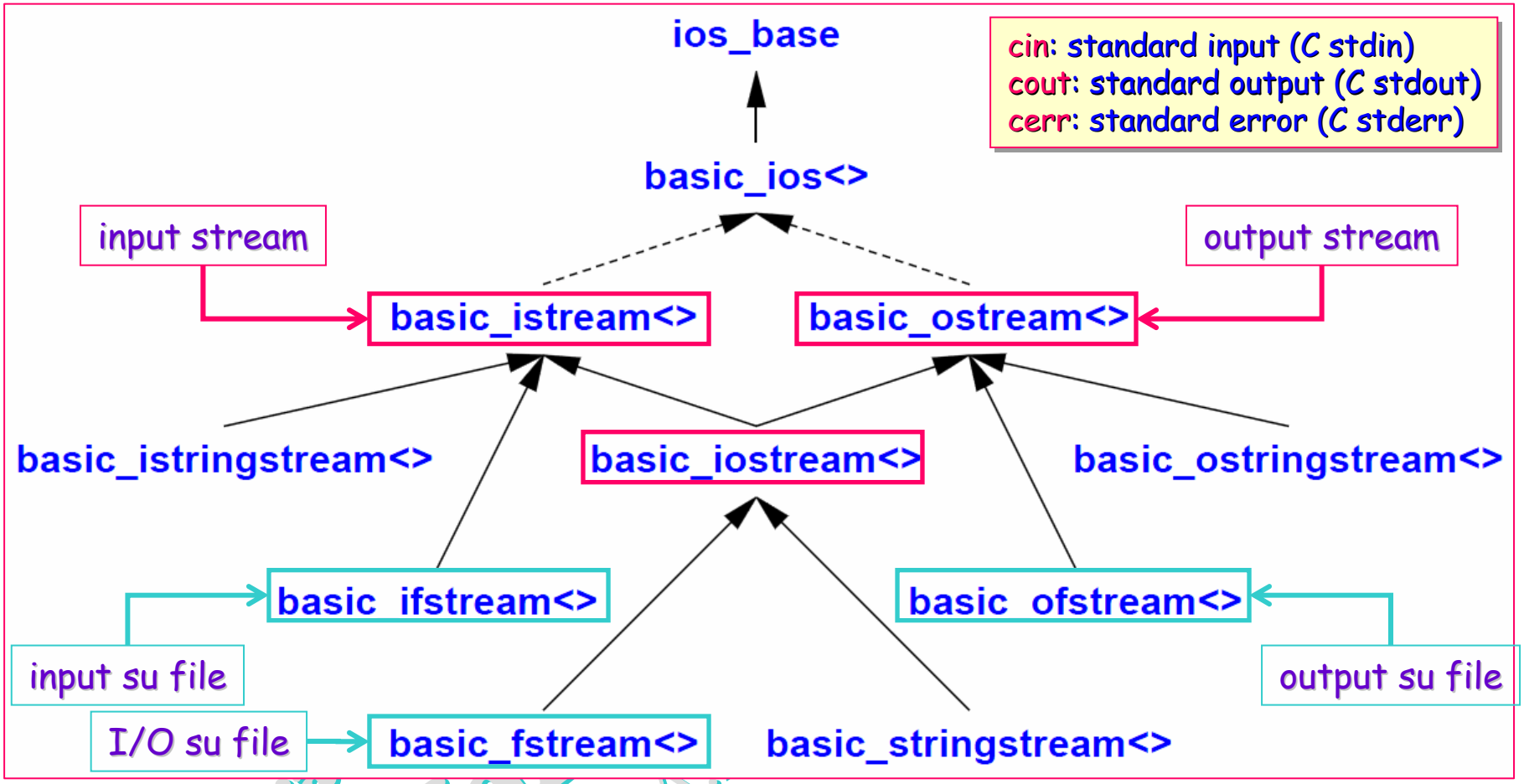
Il termine **stream** indica un generico flusso logico di dati (astrazione), sotto forma di byte, indipendente dal device coinvolto nel trasferimento (terminale, disk drive, tape drive, …). È compito del File System gestire i device associati a stream.

Text stream: una sequenza di caratteri.

Binary stream: una sequenza di byte sui quali non avviene alcuna conversione in caratteri.

Il termine **file** è invece legato al device (come ad esempio, un disk file); gli viene associato uno stream mediante l'operazione di apertura, mentre l'operazione di chiusura rimuove tale collegamento. In funzione del device il file può supportare un **indicatore di posizione** (per i disk drive si, per una stampante no!)

Gerarchia delle classi I/O Stream



Un istream può essere associato ad un input device (es.: tastiera), un file o una stringa.
Un ostream può essere associato ad un output device (finestra di testo), un file o una stringa.

I/O formattato

Format flag

adjustfield	basefield	boolalpha	dec
fixed	floatfield	hex	internal
left	oct	right	scientific
showbase	showpoint	showpos	skipws
unitbuf	uppercase		

basefield:
oct, dec, hex

floatfield:
scientific, fixed

adjustfield:
left, right, internal

Esempio 1

```
#include <iostream>
using namespace std;
int main()
{
    bool t = true;
    cout << "t=" << t << endl;
    cout << 100.0 << endl << endl;
    cout.setf(ios::boolalpha);
    cout.setf(ios::showpoint);
    cout.setf(ios::showpos); // cout.setf(ios::boolalpha | ios::showpoint | ios::showpos);
    cout << "t=" << t << endl;
    cout << 100.0 << endl;
    cout.unsetf(ios::boolalpha | ios::showpoint | ios::showpos);
    return 0;
}
```

```
t=1
100

t=true
+100.00
```

I/O formattato

Funzioni membro

streamsize width(streamsize w); — ampiezza del campo
streamsize precision(streamsize p); — numero di cifre da visualizzare
char fill(char ch); — riempimento del campo con un carattere

Esempio 2

```
#include <iostream>
using namespace std;
int main()
{
    cout.precision(4);
    cout.width(10);
    cout << 10.12345 << "\n";
    cout.fill('*');
    cout.width(10);
    cout << 10.12345 << "\n";
    cout.width(10);
    cout << "Hi!" << "\n";
    cout.width(10);
    cout.setf(ios::left);
    cout.precision(6);
    cout << 10.12345 << "\n";
    return 0;
}
```

```
  10.12
*****10.12
*******Hi!
10.1235***
```

I/O formattato: manipolatori

Manipulator	Purpose	Input/Output
boolalpha	Turns on **boolapha** flag.	Input/Output
dec	Turns on **dec** flag.	Input/Output
endl	Output a newline character and flush the stream.	Output
ends	Output a null.	Output
fixed	Turns on **fixed** flag.	Output
flush	Flush a stream.	Output
hex	Turns on **hex** flag.	Input/Output
internal	Turns on **internal** flag.	Output
left	Turns on **left** flag.	Output
nobooalpha	Turns off **boolalpha** flag.	Input/Output
noshowbase	Turns off **showbase** flag.	Output
noshowpoint	Turns off **showpoint** flag.	Output
noshowpos	Turns off **showpos** flag.	Output

Manipulator	Purpose	Input/Output
noskipws	Turns off **skipws** flag.	Input
nounitbuf	Turns off **unitbuf** flag.	Output
nouppercase	Turns off **uppercase** flag.	Output
oct	Turns on **oct** flag.	Input/Output
resetiosflags (fmtflags *f*)	Turn off the flags specified in *f*.	Input/Output
right	Turns on **right** flag.	Output
scientific	Turns on **scientific** flag.	Output
setbase(int *base*)	Set the number base to *base*.	Input/Output
setfill(int *ch*)	Set the fill character to *ch*.	Output
setiosflags(fmtflags *f*)	Turn on the flags specified in *f*.	Input/output
setprecision (int *p*)	Set the number of digits of precision.	Output
setw(int *w*)	Set the field width to *w*.	Output
showbase	Turns on **showbase** flag.	Output
showpoint	Turns on **showpoint** flag.	Output
showpos	Turns on **showpos** flag.	Output
skipws	Turns on **skipws** flag.	Input
unitbuf	Turns on **unitbuf** flag.	Output
uppercase	Turns on **uppercase** flag.	Output
ws	Skip leading white space.	Input

Esempio 3: uso dei manipolatori

Per usare i manipolatori che accettano un parametro, bisogna includere <iomanip>.

Esempio 3a

```
#include <iostream>
using namespace std;
int main()
{
    cout.setf(ios::hex, ios::basefield);
    cout << 100 << "\n"; // 100 in hex
    cout.fill('?');
    cout.width(10);
    cout << 2343.0;

    return 0;
}
```

```
64
??????2343
```

Esempio 3b

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    cout << hex << 100 << endl;
    cout << setfill('?') << setw(10)
         << 2343.0;

    return 0;
}
```

```
64
??????2343
```

File in C++

Prima di aprire un file bisogna creare uno stream. Esistono 3 tipi di stream:

Input: `ifstream in;`
Output: `ofstream out;`
Input/Output: `fstream io;`

In C++ si apre un file collegandolo ad uno stream. La funzione open() è un metodo della classe ifstream, ofstream oppure fstream. I suoi prototipi sono:

```
void ifstream::open(const char *filename, ios::openmode mode = ios::in);
void ofstream::open(const char *filename, ios::openmode mode = ios::out);
void fstream::open(const char *filename, ios::openmode mode = ios::in
                                                            | ios::out);
```

altri openmode

ios::app	append
ios::ate	seek end of file
ios::binary	file binario
ios::in	per leggere
ios::out	per scrivere
ios::trunc	sovrascrive file esistente

Esempi: sono equivalenti

```
ofstream out; out.open("test",ios::out);
```

```
ofstream out; out.open("test");
```

```
ofstream out("test");
```

Le classi ifstream, ofstream e fstream hanno costruttori che automaticamente aprono il file nella modalità default.

Esempio 4: scrittura di un file di caratteri

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

using namespace std;

int main()
{
    ofstream out("mystream.txt");

    if ( !out )
    {
        cerr << "Errore in apertura file\n";
        exit(1);
    }

    out << "Nel mezzo del cammin di nostra vita\n";
    out << "mi ritrovai per una selva oscura\n";
    out << "che la diritta via era smarrita!\n";

    out.close();
    return 0;
}
```

crea lo stream di output e gli associa il file

controlla se il file è stato aperto

scrive nel file

chiude il file

Esempio 5: lettura di un file di caratteri

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

using namespace std;

int main()
{
    ifstream in("mystream.txt");

    if ( !in )
    {
        cerr << "Errore in apertura file\n";
        exit(1);
    }
    string str;
    while ( in )
    {
        getline(in, str);
        cout << str << endl;
    }

    in.close();
    return 0;
}
```

crea lo stream di input e gli associa il file

controlla se il file è stato aperto

legge dal file finché non è finito

chiude il file

Esempio 6: scrittura di un file binario

```
#include <iostream>
#include <ostream>

using namespace std;

int main()
{
    ofstream out("mystream.bin", ios::out | ios::binary);

    if ( !out )
    {
        cerr << "Errore in apertura file\n";
        exit(1);
    }

    int vec[] = {0,1,2,3,4,5,6,7,8,9};

    out.write((char*)vec, sizeof(vec));

    out.close();
    return 0;
}
```

crea lo stream di output e gli associa il file

controlla se il file è stato aperto

scrive nel file

```
for (int k=0; k<10; k++)
    out.write((char*)&vec[k], sizeof(int));
```

chiude il file

Esempio 7: lettura di un file binario

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

using namespace std;

int main()
{
    ifstream in("mystream.bin", ios::in | ios::binary);

    if ( !in )
    {
        cerr << "Errore in apertura file\n";
        exit(1);
    }
    int dato;
    while ( !in.eof() )
    {
        in.read((char*)&dato, sizeof(dato));
        cout << dato << endl;
    }

    in.close();
    return 0;
}
```

crea lo stream di input e gli associa il file

controlla se il file è stato aperto

legge dal file finché non è finito

chiude il file