Gestione File in C++

1. Gestione dei dati permanenti: file
2. Oggetti fstream e loro uso

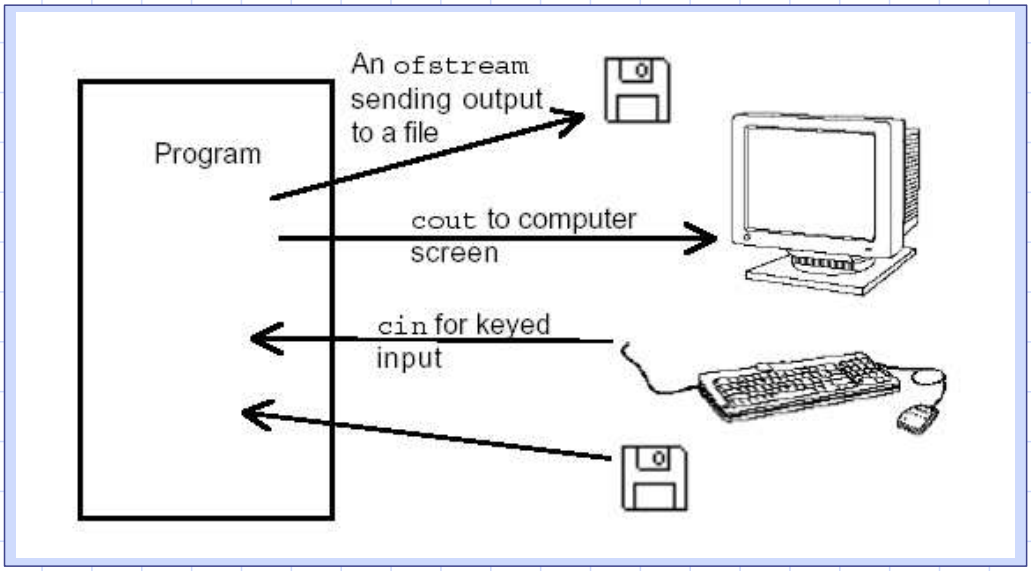
Gestione dati permanenti: *file*

- Esempi realistici di programmi trattano grandi quantità di dati. E tali dati devono essere, in genere, mantenuti in modo persistente.
- La gestione dei dati persistenti prevede la creazione di infrastrutture per la memorizzazione ed il reperimento degli oggetti
- A livello del sistema si ricorre a DBMS (Data Base Management System). L'uso di DBMS esula dall'ambito del corso … In casi particolari, l'uso di file può sostituire l'uso di database

Allora:

È necessario dare una veloce introduzione alla manipolazione di file in C++.

Programma e dispositivi input/output



Oggetti filestream

- Se un programma ha bisogno di elaborare dati da/per file, deve dichiarare oggetti di tipo "filestream".
- La libreria <fstream> contiene le definizioni per tali oggetti.
- Gli oggetti filestream sono di tre tipi:
  - ifstream --- per leggere dati da file (input file stream)
  - ofstream --- per scrivere dati su file (output file stream)
  - fstream --- per leggere e scrivere dati (in differenti momenti) su file

Definire un filestream

```
#include <fstream> // file intestazione da includere per i filestream
int main()
{
   ifstream input("theinput.txt", ios::in); // costruttore
   ...
```

```
#include <fstream>
void main()
{
   ifstream input;  // altro costruttore. L'oggetto filestream input non
   ...                    // è collegato a nessun file
   input.open("file1.txt", ios::in); // uso metodo open ...
   ...
```


Lettura da ifstream

Una volta creato un oggetto filestream di input, può essere usato come l'oggetto cin …

```
#include <fstream>
void main()
{
   ifstream input("theinput.txt", ios::in);
   long l1, l2; double d3; char ch;
   ...
   input >> d3;  // read a double from file
   ...
   input >> ch;  // read a character
   ...
   input >> l1 >> l2;  // read two long integer
   ...
}
```

Scrittura su ofstream

Allo stesso modo, oggetti ofstream possono essere usati come cout:

```
#include <fstream>
void main()
{
  ofstream out1("results.txt", ios::out); // create an output file
  int i; double d;
  ...
  out1 << "The results are :" << endl; ... // send header to file
  for(i=0;i<100; i++) {
  ...
    out1 << "i : " << i << ", di " << d << endl; // send data to file
  ...
}
out1.close();  // finished, close the file.
```


Lo stato degli fstream

Supponendo di aver dichiarato
ifstream in1("indata.txt", ios::in);
si può testare lo stato di in1 con i metodi:

in1.good() *returna "true" se in1 è OK per l'uso*
in1.bad() *returna "true" se l'ultima operazione su in1 è fallita e non c'è modo di recuperare dal fallimento*
in1.fail() *returna "true" se l'ultima operazione su in1 è fallita e c'è modo di recuperare dal fallimento*
in1.eof() *returna "true" se non ci sono altri dati da leggere*

Opzioni di apertura

ios::in	aperto per la lettura
ios::out	aperto per la scrittura
ios::ate	posiziona alla fine del file
ios::app	apre il file in modalità “append”
ios::trunc	tronca il file in apertura
ios::nocreate	se il file non esiste non cerca di crearlo
ios::noreplace	l’apertura fallisce se il file esiste
ios::translate	converte CR/LF in newline in input; il viceversa in output

Combinazioni di opzioni

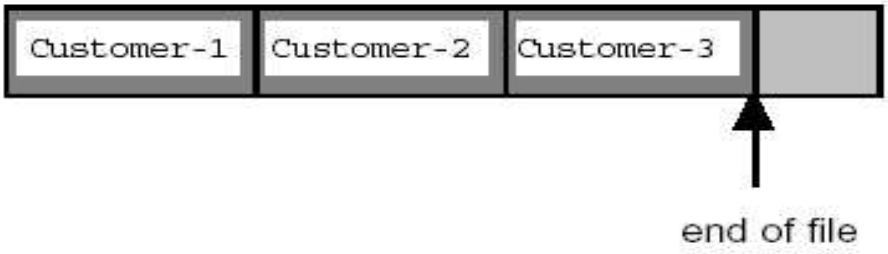
Combinationi tipiche per l’apertura sono:

ios::in | ios::nocreate	*apertura se il file esiste, fallimento altrimenti*
ios::out | ios::noreplace	*apre un nuovo file in output, fallimento se il file già esiste*
ios::out | ios::ate	*(ri)apre in output un file esistente, aggiunge i dati alla fine, dopo quelli già presenti*
ios::out | ios::noreplace | ios::translate	*apre nuovo file, fallisce se il file già esiste, traduce i caratteri di newline*
ios::out | ios::nocreate | ios::trunc	*apre un nuovo file in input, fallimento se il file non esiste, elimina dati precedentemente contenuti*

File di record e file di testo

A

"Record structured" file:



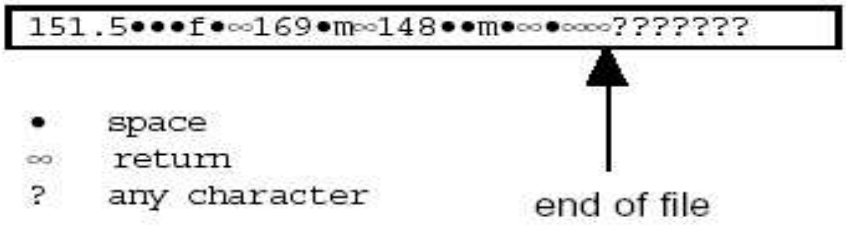
B

Text file:

Input text

```
151.5  f
169 m
148 m
```

Actual input file



Esempio problematico

```
ifstream kids("PesoBambini.dat", ios::in | ios::nocreate);

if(!kids.good()) {
    cout << "File non esistente!" << endl;
    exit(1);
}
while(! kids.eof()) {
    double height;
    char gender_tag;
    kids >> height >> gender_tag;
    …
}
```

Possibilità di loop infinito (confronta con l'input nella figura precedente)