Università della Calabria
Corso di Laurea in Ingegneria Elettronica

Programmazione Orientata agli Oggetti
A.A. 2006/2007

Input/Output

Paolo Trunfio *

* DEIS, Università della Calabria – http://si.deis.unical.it/~trunfio

Gli stream

Il sistema di I/O di C++ è basato sul concetto di *stream*. Uno stream è un'interfaccia logica per leggere (o scrivere) dati da un dispositivo del computer (ad es.: tastiera, monitor, file, porte, etc.).

C++ definisce alcuni stream predefiniti, tra i quali *cin* e *cout* per la lettura da tastiera e la scrittura su monitor.

Il programmatore può definire altri stream, ad esempio per leggere e scrivere su file.

Una caratteristica fondamentale degli stream è che le loro funzionalità sono indipendenti dal dispositivo ai quali sono collegati.

Ad esempio, è possibile scrivere (o leggere) su un file con la stessa sintassi con cui si scrive (o legge) sul monitor.

Gli strumenti principali per manipolare gli stream sono l'operatore di inserimento **<<**, che consente di inserire caratteri in uno stream, e l'operatore di estrazione **>>**, che consente di estrarre caratteri da uno stream.

Manipolatori di I/O

C++ fornisce un insieme di *manipolatori* di I/O per controllare la formattazione di uno stream. Un esempio di manipolatore di output è *endl*:

```
cout << "una linea" << endl;  // inserisce un carattere di fine linea
```

Esistono manipolatori che non accettano argomenti, come *endl*, ed altri che accettano parametri.

Ad esempio, il manipolatore *setprecision*(int *p*) imposta la precisione con la quale un numero deve essere visualizzato:

```
#include <iomanip>  // necessario per i manipolatori con argomenti

...

double p = 3.575;

cout << p << endl;  // 3.575

cout << setprecision(2) << p << endl;  // 3.6
```


Manipolatori di I/O

Esempi di manipolatori di output sono *dec*, *hex*, *oct*, *setw*(int *w*):

```
int n = 75;

cout << hex << n << "\t" << oct << n << endl;  // 4b    113

cout << dec << n << endl;  // 75

cout << setw(6) << "POO" << endl;  // Ampiezza di campo = 6: "      POO"
```

Esempi di manipolatori di input sono *dec*, *hex*, *oct*:

```
int n;

cout << "Inserisci esadecimale: ";

cin >> hex >> n;  // per es: 4b

cout << "Decimale corrispondente = " << dec << n << endl;  // 75
```

Le funzioni get, put e getline

La funzione *get* legge un carattere per volta da uno stream:

```
char c = cin.get();
```

La funzione *put* scrive un carattere per volta su uno stream:

```
char c = 'x';
cout.put(c);
```

La funzione *getline* legge un'intera linea da uno stream e la copia nella stringa ricevuta per riferimento:

```
string s;
getline(cin, s, '\n');  // legge dallo stream cin e scrive nella stringa s
cout << s << endl;
```

Il terzo parametro di *getline* è il carattere di terminazione della linea (in questo esempio è il "fine linea")

Lettura di una linea con get

```
#include <iostream>
using namespace std;

int main() {
  string s;
  char c;
  do {
    c = cin.get();
    if (c == '\n')
      break;
    else
      s += c;
  } while (c != '\n');
  cout << s << endl;
}
```

I/O su file

Per effettuare l'I/O su file è necessario includere il file di intestazione *fstream*:

#include <fstream>

Le istruzioni seguenti creano uno stream di input ed uno stream di output per leggere e scrivere su file:

ifstream in; // *in* è uno stream di input
ofstream out; // *out* è uno stream di output

Dopo la creazione, uno stream deve essere associato ad un file:

out.open("test.txt"); // associa lo stream di output *out* al file *test.txt*

Successivamente, è possibile scrivere (o leggere) sullo stream usando gli stessi operatori utilizzati per lo stream predefinito *cout* (o *cin*):

out << "linea di prova" << endl; // scrive "linea di prova" sul file *test.txt*

Scrittura su file

```
int main() {
  ofstream myStream;  // crea un ofstream di nome myStream
  myStream.open("prova.txt");  // associa myStream al file prova.txt
  if (!myStream)
    cout << "Errore: impossibile aprire il file\n";
  else {
    myStream << "prima riga" << endl;
    myStream << "seconda riga" << endl;
    cout << "Scrittura del file completata\n";
  }
  myStream.close();  // chiude myStream
}
```

Il programma precedente crea (o sovrascrive) il file *prova.txt* con il seguente contenuto:

prima riga
seconda riga

Scrittura su file: modalità “append”

Il programma seguente appende due righe in fondo al file specificato, se il file esiste; se il file non esiste, lo crea ed inserisce le due righe:

```
int main() {
  cout << "Nome del file: ";
  string nome;
  cin >> nome;
  ofstream file;
  file.open(nome.c_str(), ios_base::app);  // modalità append
  if (!file)
    cout << "Errore: impossibile aprire il file\n";
  else {
    file << "riga aggiuntiva 1" << endl;
    file << "riga aggiuntiva 2" << endl;
    cout << "Scrittura del file completata\n";
  }
  file.close();
}
```


Lettura da file

Legge le stringhe da un file e le stampa su video su righe separate:

```
int main() {
  cout << "Nome del file: ";
  string nome;
  cin >> nome;
  ifstream file;
  file.open(nome.c_str());
  if (!file)
    cout << "Errore: impossibile aprire il file";
  else {
    string s;
    while (file >> s)  // restituisce false se non può estrarre una stringa
      cout << s << endl;
  }
  file.close();
}
```

Lettura da file: rilevamento dell'"end-of-file"

Legge le righe da un file e le stampa su video su righe separate:

```
int main() {
  string nome;
  cout << "Nome del file: ";
  cin >> nome;
  ifstream file;
  file.open(nome.c_str());
  if (!file)
    cout << "Errore: impossibile aprire il file" << endl;
  else {
    while (!file.eof()) {  // eof restituisce true quando incontra l'end-of-file
      string s;
      getline(file, s, '\n');
      cout << s << endl;
    }
  }
  file.close();
}
```