LA GESTIONE DEI FILE DI TESTO IN C++

Le classi per la gestione dei file.

Il C++ è un linguaggio senza alcuna istruzione primitiva per la gestione delle operazioni di input/output; esse vengono implementate attraverso apposite librerie di classi che costituiscono di fatto un'estensione del linguaggio.
Queste classi sono associate al concetto di ***stream*** (flusso), che rappresenta un'astrazione dei dispositivi di ingresso/uscita ed ha il duplice scopo di rendere la scrittura dei programmi indipendente dal particolare dispositivo impiegato e di semplificare i problemi di *portabilità* dei programmi stessi.

Uno stream può essere considerato come un flusso di dati che passa sequenzialmente da una sorgente (produttore dei dati) ad una destinazione (consumatore dei dati).

Si parla quindi di ***stream di input*** quando il flusso prevede come sorgente una periferica esterna (es. la tastiera) e come destinazione la memoria, e di ***stream di output*** nel caso contrario (ad es. quando il flusso è diretto verso il video).

Le classi C++ che permettono la gestione delle operazioni di input/output sono la **istream** (per gli stream di input) e la **ostream** (per gli stream di output), dalle quali deriva poi la **iostream** (per gli stream di input/output). Quest'ultima è quella generalmente utilizzata nei nostri programmi poiché in essa sono definiti i due oggetti associati alle periferiche standard di input e di output, e cioè:

- ⇒ **cin** associato alla tastiera;
- ⇒ **cout** associato al video.

Queste classi sono utilizzabili includendo nel nostro programma il file di intestazione **iostream.h**, attraverso l'istruzione: #include <iostream.h>.

Anche i file sono visti in C++ come dispositivi di input/output e sono quindi gestiti come stream attraverso tre nuovi tipi di classi, definiti nel file di intestazione **fstream.h** e sono:

- ⇒ **ifstream** che deriva da istream e serve a gestire i file solo per operazioni di input (cioè lettura);
- ⇒ **ofstream** che deriva da ostream e serve a gestire i file solo per operazioni di output (cioè scrittura);
- ⇒ **fstream** che deriva dalle due precedenti e serve a gestire i file per operazioni di aggiornamento.

File di testo e file binari.

I file di testo sono costituiti esclusivamente da tutti quei caratteri (compreso il *blank* o spazio) che possono far parte di un testo. Essi sono generalmente composti da una sequenza di caratteri e sono organizzati a linee; ogni linea è terminata con il carattere **\n** (fine linea). Il file è terminato da un indicatore di fine file, EOF, che costituisce un carattere speciale (cod. ASCII 26, sulla tastiera ^Z). L'accesso ai file di testo è di tipo ***sequenziale*** in quanto i caratteri vengono letti o scritti in sequenza dal primo all'ultimo.
I file di testo contengono informazioni ***listabili***, cioè visualizzabili e stampabili direttamente.
I file binari sono invece costituiti da una sequenza di byte che rappresentano informazioni in formato interno, in genere associate a strutture record definite nel programma.

In questa dispensa verranno trattati soltanto i file di testo, anche se molti concetti esposti sono validi anche per i file binari.

La dichiarazione di un oggetto di tipo file.

Per dichiarare, nel nostro programma, un identificatore associato ad un file di testo utilizzeremo la seguente:

<tipo file> nome_identificatore;

dove <tipo file> sarà una delle tre classi ifstream, ofstream o fstream a seconda che il file sarà utilizzato solo in lettura, solo in scrittura o in lettura/scrittura.
Ad esempio, volendo dichiarare nel nostro programma un *oggetto* di nome *miofile* da associare ad un file di testo scriveremo:

ifstream miofile;	se il file sarà utilizzato solo in lettura
ofstream miofile;	se il file sarà utilizzato solo in scrittura
fstream miofile;	se il file sarà utilizzato solo in lettura/scrittura

Apertura del file.

L'operazione di apertura di un file serve essenzialmente per associare l'identificatore definito nel programma con le modalità sopra riportate con il file effettivamente memorizzato sul supporto fisico.

L'apertura di un file, quindi, associa uno stream (che è un elemento astratto) ad un file fisico.

Per aprire un file si usa la funzione **open()**, membro delle classi di cui sopra. La sintassi è:

nome_identificatore**.open**(nome_file_fisico [,<specificatori modalità di apertura>]);

Ad esempio, volendo aprire il file miofile prima visto in modalità di sola lettura associandolo al file su disco di nome LETTERA.TXT, scriveremo:

```
…
ifstream miofile;
miofile.open("LETTERA.TXT");
…
```

Vi è però un'altra possibilità di eseguire la stessa operazione, dichiarando cioè miofile di tipo **fstream** e specificando la modalità di apertura del file:

```
…
fstream miofile;
miofile.open("LETTERA.TXT",ios::in);
…
```

ios::in è un enumeratore predefinito che specifica la modalità di apertura del file in sola lettura.
Di seguito sono elencati i più importanti enumeratori predefiniti per l'apertura di un file in C++:

ios::in	apre il file in sola lettura; se il file non esiste l'apertura dà errore;
ios::out	apre il file in sola scrittura; se il file non esiste viene creato, se esiste viene cancellato e sovrascritto;
ios::app	apre il file in modalità *append*: i nuovi dati sono aggiunti alla fine del file;

ios::trunc se il file esiste viene cancellato e sovrascritto;
viene utilizzato in abbinamento a ios::in o a ios::app;

ios::nocreate in scrittura apre il file solo se esiste, se no dà errore;

ios::noreplace in scrittura apre il file solo se non esiste, se no dà errore.

Per specificare modalità di apertura differenti è possibile combinare gli enumeratori sopra elencati attraverso l'operazione di *bitwise OR* (simbolo |).
Ad esempio, l'enumeratore **ios::in | ios::out** specifica una modalità di apertura del file sia in lettura che in scrittura.
L'uso degli specificatori della modalità di apertura è indispensabile per i file dichiarati di tipo **fstream**.
Per un file dichiarato di tipo **ifstream**, infatti, non è necessario specificare la modalità di apertura poiché si ha già per default la modalità **ios::in**.
Analogamente un file dichiarato di tipo **ofstream** sarà di default aperto in modalità **ios::out**.
Per **fstream**, invece, non vi è default e pertanto il modo va esplicitato.

Vediamo ora come le due operazioni di dichiarazione e di apertura di un file possano essere riunite in un'unica istruzione. A tal fine dichiariamo e apriamo lo stesso file visto negli esempi precedenti:

```
…
fstream miofile("LETTERA.TXT", ios::in);
…
```

Questa istruzione è del tutto equivalente alle due viste in precedenza:

```
…
fstream miofile;
miofile.open("LETTERA.TXT",ios::in);
…
```

Come si può osservare non viene più utilizzata la funzione **open()** per l'apertura del file, che sembra essere fatta implicitamente all'atto della dichiarazione. In realtà l'apertura del file è eseguita dal ***metodo costruttore*** della classe **fstream** che viene richiamato automaticamente quando ***istanziamo*** (cioè dichiariamo) il nuovo oggetto miofile di questa classe (questi concetti possono essere chiari solo a coloro che possiedono nozioni di programmazione ad oggetti).

Errore nell'apertura di un file.

Vi sono diversi metodi per verificare, all'interno del programma, se l'operazione di apertura di un file sia andata a buon fine o se abbia invece generato un errore.
Un primo metodo utilizza il nome stesso del file. Infatti se l'apertura del file avviene senza errori al nome del file è associato un numero diverso da zero. Questo perché, come si diceva prima, l'identificatore del file è a tutti gli effetti l'identificatore di un oggetto (di classe fstream, ifstream o ofstream). Ebbene l'identificatore di un oggetto può essere considerato equivalente ad un puntatore che, se l'oggetto viene correttamente istanziato, conterrà l'indirizzo della locazione di memoria a partire dalla quale sono memorizzati gli attributi dell'oggetto stesso. Per lo stesso motivo se l'apertura del file non viene completata al suo identificatore è associato il valore zero (perché non esiste l'indirizzo di memoria a partire dal quale sono memorizzati gli attributi dell'oggetto).
Pertanto un esempio di codice che verifica la corretta apertura di un file può essere il seguente:

```
…
fstream miofile(“LETTERA.TXT”, ios::in);

if (!miofile)
   {
        cout << “Errore nell’apertura del file” << endl;
   }
…
```

Un altro metodo (del tutto analogo al precedente) consiste nell’utilizzo della funzione **is_open()**. Questa torna zero se non è stato possibile associare uno stream ad un file aperto su disco, altrimenti ritorna un valore diverso da zero. Vediamone un esempio di utilizzo con lo stesso file dell’esempio precedente:

```
…
if (!miofile.isopen())
   {
        cout << “Errore nell’apertura del file” << endl;
   }
…
```

Un’altra possibilità è offerta dalla funzione **fail()**, che invece assume il valore ***true*** quando un’operazione sul file produce un errore:

```
…
if (miofile.fail())
   {
        cout << “Errore nell’apertura del file” << endl;
   }
…
```

La chiusura di un file.

Quando non è più necessario accedere ad un file, questo deve essere chiuso in modo da rendere disponibile a un nuovo uso lo stream cui era stato associato. Molti sistemi operativi, infatti, limitano il numero massimo di file che possono essere contemporaneamente aperti e pertanto è buona norma chiudere i file di cui non si ha più bisogno.
L’operazione in esame si realizza semplicemente con l’istruzione:

nome_identificatore**.close()**;

Riferendoci all’esempio utilizzato sino ad ora, per chiudere il file miofile scriveremo:

miofile**.close()**.

Raggiungimento della fine del file.

Per verificare, durante la fase di lettura di un file, se si è giunti all’ultimo dato memorizzato si utilizza la funzione **eof()**. Questa ritorna il valore ***true*** se si è giunti alla fine del file, ***false*** in caso contrario.
La sintassi è sempre:

nome_identificatore.eof()

Lettura scrittura di un carattere.

Per leggere un carattere dal file si utilizza la funzione **get()**, che restituisce il carattere letto oppure l'informazione di EOF (End Of File).
Per scrivere un carattere in un file si utilizza la funzione **put(char c)**.
L'esempio di codice che segue può chiarire l'utilizzo di queste due funzioni (oltre che di tutto il resto).
Si tratta di un programma che esegue la copia di un file su di un altro:

```
#include <stdlib.h>
#include <iostream.h>
#include <fstream.h>

void main()
{
        char c;

        // Apertura dei file
        ifstream sorg("ORIGINE.TXT");
        ofstream dest("COPIA.TXT");

        if (sorg && dest)       {                       // Verifica la corretta apertura dei file

                while (!sorg.eof())     {               // Verifica di non aver raggiunto la fine
                                                           del file di partenza
                        c = sorg.get();                 // Legge un carattere dal file origine
                        dest.put(c);                    // Scrive il carattere appena letto sul
                                                           file destinazione
                }

                // Chiusura dei file
                sorg.close();
                dest.close();

        }

        else    {                                       // Stampa messaggio di errore nell'apertura
                cout << "Errore nell'apertura del file" << endl;
        }

}
```

Esercizi proposti.

Lo studente sviluppi un'applicazione C++ che trascriva un file di testo su di un altro file trasformando tutte le lettere minuscole in maiuscole. I nomi dei due file devono essere entrambi specificati in input. Se il file di partenza non esiste, deve essere visualizzato un apposito messaggio d'errore. Per ottimizzare le operazioni di lettura/scrittura si utilizzi un *buffer* di memoria da 1 Kbyte.