Streams

Antonio Espín Herranz

Flujos de Entrada / Salida

- C++ da un tratamiento a los ficheros como flujos de información (**stream**).
- Para C++ puede ser un flujo de bytes que vienen de un fichero o que vienen de un string o de un dispositivo.
- Dispone de distintas clases para ello.

Para procesar ficheros

- Permite acceso **secuencial** y **directo** a los ficheros.
  - Implementado por nosotros podemos hacer un acceso **indexado**.
- Permite procesar ficheros de:
  - **Texto** y **binarios**.
- Permite leer/escribir ficheros a nivel de:
  - **Carácter**, **cadenas** de caracteres y **registros**.

Para procesar un fichero

LECTURA DE UN FILE

Abrir un flujo desde un fichero.

Mientras haya información

    Leer información

Cerrar el flujo

ESCRITURA DE UN FILE

Abrir un flujo hacia un fichero.

Mientras haya información

    Escribir información

Cerrar el flujo.

Para procesar ficheros

- **fstream**.
  - Para leer y escribir ficheros. Habrá que indicar el modo a la hora de escribir o leer un fichero.
- **ifstream**
  - Para leer un fichero
- **ofstream**
  - Para escribir un fichero.

Para procesar string

- Las clases istringstream, ostringstream y stringstream permiten definir flujos de entrada, salida y entrada / salida.

Ejemplo

- Con la clase **istringstream** podemos dividir una cadena en tokens a partir de un carácter de separación:
- Implementación de Split, se pueden pasar los tokens a un vector:

Ejemplo

- **Separamos in string por la coma: ‘,’**

```
string input = "abc,def,ghi";
istringstream ss(input);
string token;
while(getline(ss, token, ','))
    cout << token << endl;
```

Ejemplo para escribir fichero txt

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

int main(){
    ofstream ofs;
    // Abrir el fichero de texto para escribir
    ofs.open(“fichero.txt”, ios::out);
    ofs << “Hola mundo\n”;
    ofs << 9 << “ * “ << 256 << “ = “ << 9*256 << “\n”;
    ofs.close();
    return 0;
}
```

Ejemplo para escribir fichero bin

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

int main(){
  int n = 0;
  char *s = "¡Hola mundo!\n";
  ofstream ofs; // flujo
  // Abrir el fichero
  ofs.open("mifichero-b.bin", ios::out|ios::binary);  // fichero binario
  // Escribir en el fichero
  ofs.write(s, strlen(s));
  n = 9;        ofs.write((char *)&n, sizeof(int));
  n = 256;      ofs.write((char *)&n, sizeof(int));
  n = 17432583; ofs.write((char *)&n, sizeof(int));
  // Cerrar el fichero
  ofs.close();
  return 0;
}
```

La clase streambuf

- Esta clase es abstracta se derivan dos clases: filebuf y stringbuf.
- Ambas proporcionan un buffer, una para ficheros y la otra para string.
- Podemos utilizar la clase filebuf para abrir ficheros. Proporciona búferes para gestionar la E/S sobre ficheros.

La clase filebuf

- Tenemos que incluir el fichero de librería:
  #include <fstream>

- Tenemos el método open para abrir un fichero.
  filebuf *open(char *nombreFich, ios::openmode modo);
  - Este método devuelve 0 si no se ha podido abrir el fichero.

- El método is_open() nos devuelve true si el fichero está abierto.

Los modos de apertura

- Está definidos mediante constantes. Para combinar mas de una constante utilizamos **|**

- in, out, trunc, app, binary:
  - La forma de utilizarlas: ios::out **|** ios::binary

Equivalencia de modos en C

- Fichero de Texto:
  - out — w — escribir
  - out | app — a — añadir
  - out | trunc — w — escribir
  - in — r — leer
  - in | out — r+ — leer y escribir
  - In | out | trunc — w+ — escribir y leer

- Fichero Binario:
  - Serían los mismo flag pero añadiendo | binary.

La clase ostream

- Proporciona la funcionalidad necesaria para acceder de forma secuencial o aleatoria de un fichero para escribir.
- La clase ostream trabaja con filebuf.
- #include <ostream>
- Normalmente esta clase no se utiliza tenemos **ofstream** para ficheros y **ostringstream** para cadena de caracteres.

Ejemplo de ostream / filebuf

```
#include <ostream>
#include <fstream>

filebuf buf;
buf.open(“fichero”, ios:out);
// Se liga el búfer con el flujo en el constructor:
ostream os(&buf);
os << “Datos a escribir” << endl;
```

La clase istream

- Proporciona la funcionalidad necesaria para acceder de forma secuencial o aleatoria de un fichero para leer.
- La clase istream trabaja con filebuf.
- #include <istream>
- Normalmente vamos a trabajar con la clase **ifstream** para ficheros y **istringstream** para cadena de caracteres.

Ejemplo de istream / filebuf

```
#include <istream>
#include <fstream>

filebuf buf;
buf.open(“fichero”, ios::in);
// Se liga el búfer con el flujo en el constructor:
istream is(&buf);
string linea;
getline(is, linea);
cout << “Datos leidos: “ << linea << endl;
```

Clase ofstream

- Hereda de ostream, esta clase conecta de forma automática el flujo con el buffer (filebuf).

- #include <fstream>
  - En el constructor le pasamos los parámetros:
  - ofstream(const char *, modo);
  - Devuelve 0 si no puede abrir el fichero.
  - close(): cierra el fichero.
  - is_open() : ¿Está abierto el fichero?
  - Por defecto, abre para escritura.
  - filebuf *rdbuf(): Devuelve un puntero al búfer asociado.

Clase ifstream

- Hereda de istream, esta clase conecta de forma automática el flujo con el buffer (filebuf).

- #include <fstream>
  - En el constructor le pasamos los parámetros:
  - ifstream(const char *, modo);
  - Devuelve 0 si no puede abrir el fichero.
  - close(): cierra el fichero.
  - is_open() : ¿Está abierto el fichero?
  - Por defecto, abre para lectura.
  - filebuf *rdbuf(): Devuelve un puntero al búfer asociado.

Clase fstream

- Deriva de iostream especializada en manipular ficheros abiertos en modo lectura / escritura.
- Se asocia automáticamente con el búfer como sucede con las dos clases anteriores.
- #include <fstream>
- El constructor es como antes, por defecto abre para in | out. Si el fichero existe no se destruye, y si no existe se producirá un error.
- También soporta los métodos de las dos anteriores.

Ejemplos

- **// Abrir para escribir:**
  - ofstream os(“texto.txt”)
  - if (!os)
    - throw “No se puede abrir el fichero”;
- **// Abrir para leer:**
  - ifstream is(“texto.txt”)
  - if (!is)
    - throw “No se puede abrir el fichero”;
- **// Abrir para escribir y leer:**
  - fstream fs(“texto.txt”);
  - if (!fs)
    - throw “No se puede abrir el fichero”;

E/S Carácter a Carácter

- Se utilizan los métodos put y get sobre flujos de salida y entrada respectivamente.

- Podemos chequear el estado del flujo mediante los métodos: good(), fail(), bad() y eof().

Ejemplo

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

using namespace std;

int main(){

    fstream fuente, destino;

    fuente.open("entrada.txt", ios::in);
    if (!fuente){
    cout << "No se encontró el fichero .." << endl;
    exit(-1);

    }

    destino.open("salida.txt", ios::out);
    if (!destino){
    cout << "Error al abrir el fichero destino ..." << endl;
    exit (-1);
    }
    char car;

    while (fuente.get(car))
        destino.put(car);

    // Chequeo para saber si se ha procesado el fichero de entrada entero:
    if (!fuente.eof())
    cout << "Error al procesar la entrada" << endl;

    cout << "Se ha procesado el fichero: " << endl;

    fuente.close();
    destino.close();
    return 0;
}
```

E/S Cadenas de caracteres

- Los datos se pueden escribir o leer utilizando los métodos operator<< o getline, respectivamente.

- Podemos leer cadenas:

  string cadena;

  getline(cin, cadena);

- Para escribir:

  fstream os(“fichero”, ios::out);

  os << cadena << endl;

E/S registros

- Para trabajar con registros utilizaremos los métodos: read / write heredados de fstream.
- Registros conjunto de datos de longitud fija.

```
ofs.write(reinterpret_cast<char *>(&persona),
   sizeof(persona));
ifs.read(reinterpret_cast<char *>(&persona),
   sizeof(persona));
```

Ejemplo: escribir registros

```
struct t_tfno {   char nombre[30];   char direccion[40];   long telefono; };

 t_tfno persona; // estructura de tipo t_tfno
int tam = sizeof(t_tfno);
// Abrir el fichero para escribir
ofstream ofs("lista.tfno");
// Leer los datos
cout << "Nombre:    "; cin.getline(persona.nombre, 30);
cout << "Dirección: "; cin.getline(persona.direccion, 40);
cout << "Teléfono:  "; cin >> persona.telefono;
// Escribir el tamaño del registro y su contenido
ofs.write(reinterpret_cast<char *>(&tam), sizeof(int));
ofs.write(reinterpret_cast<char *>(&persona), sizeof(persona));
// Cerrar el fichero
ofs.close();
```

Ejemplo: leer registros

```
t_tfno persona; // estructura de tipo t_tfno
int tam = sizeof(t_tfno);
// Abrir el fichero para leer
ifstream ifs("lista.tfno");
// Leer el tamaño del registro y su contenido
ifs.read(reinterpret_cast<char *>(&tam), sizeof(int));
ifs.read(reinterpret_cast<char *>(&persona),
sizeof(persona));
// Mostrar los datos leídos
cout << "Tamaño:    " << tam << endl;
cout << "Nombre:    " << persona.nombre << endl;
cout << "Dirección: " << persona.direccion << endl;
cout << "Teléfono:  " << persona.telefono << endl;
// Cerrar el fichero
ifs.close();
```