

Metodología de la Programación

Tema 4.- Flujos asociados a ficheros (fstream)

Introducción

Tipos de ficheros

Operaciones específicas



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 1

Introducción

Relación entre flujos y ficheros

Fichero: secuencia de bytes almacenados en un dispositivo de almacenamiento masivo (disco duro, CD, ...). Esta secuencia acaba con el carácter **EOF** (*End Of File*).

El objetivo de los ficheros es que la información permanezca cuando se desconecta la alimentación del ordenador.

En C++, para poder acceder a un fichero lo que se hace es **asociarle un flujo**. De esta forma nosotros lo que hacemos es trabajar con el flujo (y este ya se encargará de transferir la información a o desde el fichero).

Al gestionar el fichero como si fuera un flujo *podemos usar todas las operaciones* que hemos visto sobre flujos.



- Operaciones de E/S
- Manipuladores de formato
- Operadores de posicionamiento
- ...

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 2

Introducción

Más sobre ficheros

Una de las diferencias más importantes entre los flujos por defecto (*cin*, *cout*, ...) y los ficheros es que, mientras que aquellos se crean de forma automática al incluir *iostream*, los ficheros hemos de crearlos nosotros:

- *cin* siempre está asociado al teclado.
- *cout* siempre está asociado al monitor.
- Un flujo de tipo fichero puede estar asociado a distintos ficheros por lo que somos nosotros los que decidimos con qué fichero lo queremos asociar.

Para poder usar ficheros hemos de incluir el fichero **fstream**

Las *clases* para manipular ficheros son:

ifstream	→ Flujo asociado a ficheros de E/
ofstream	→ Flujo asociado a ficheros de /S
fstream	→ Flujo asociado a ficheros de E/S

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 3

Introducción

Jerarquía de clases

Jerarquía de clases que relaciona los flujos vistos hasta ahora:

```

graph TD
    istream[istream] --> ifstream[ifstream]
    istream --> iostream[iostream]
    ostream[ostream] --> iostream
    ostream --> ofstream[ofstream]
    fstream[fstream] --> iostream
  
```

Cualquier cosa que haga con **istream** se puede hacer con **iostream**
 Cualquier cosa que haga con **istream** se puede hacer con **ifstream**
 Cualquier cosa que haga con **ostream** se puede hacer con **iostream**
 Cualquier cosa que haga con **ostream** se puede hacer con **ofstream**
 Cualquier cosa que haga con **iostream** se puede hacer con **fstream**
Por transitividad:
 Cualquier cosa que haga con **istream** se puede hacer con **fstream**
 Cualquier cosa que haga con **ostream** se puede hacer con **fstream**

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 4

Metodología de la Programación

Tema 10.- Flujos asociados a ficheros (fstream)

- Introducción**
- Tipos de ficheros**
- Operaciones específicas**



UGR Universidad de Granada DECSAI Universidad de Granada

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 5

Tipos de ficheros Clasificación

Tipos de ficheros

$\left\{ \begin{array}{l} \text{de entrada} \\ \text{de salida} \\ \text{de entrada/salida} \end{array} \right.$	$\left\{ \begin{array}{l} \text{de texto (ASCII)} \\ \text{binarios (sin formato)} \end{array} \right.$
--	---

La información **siempre** se almacena en binario (secuencias de 0, 1)

La podemos interpretar de distintas formas

Si encontramos la secuencia **0100 0001**:

- Podemos interpretarla como un número entero → **65**
- Podemos interpretarla como el código ASCII de un símbolo → **A**

El número **65** lo podemos almacenar como:

- La secuencia binaria que representa su magnitud: **0100 0001**
- La secuencia de símbolos ASCII 6 y 5. En este caso guardaremos el código ASCII del símbolo '6' (**0011 0110**) seguido del código ASCII del símbolo '5' (**0011 0101**)

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 6

Tipos de ficheros

Ficheros de texto y binarios

Ficheros de texto (ASCII)

Cada byte que hay almacenado *debe interpretarse* como un número que representa la magnitud de un **código ASCII**.

Ejemplo: Si escribimos un número se escribirán los códigos ASCII de todos los dígitos o símbolos que forman el número.

6.32 → 00110110 00101110 00110011 00110010

Ficheros binarios

Se ven como una secuencia de bytes que **no** tienen porque tener ninguna relación con los símbolos ASCII.

El *significado* de la secuencia de bytes vendrá determinado por el programa que lee o escribe el fichero.

Cuando escribimos un dato lo que se almacena es la **representación interna del dato** (tal cual está en memoria).

Ejemplo: tenemos un ordenador que usa la representación *IEEE Std 754* de 4 bytes para los números reales:

6.32 → 0 10000001 1001010001111010111001

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 7

Tipos de ficheros

Ficheros de texto y binarios

Los ficheros de texto se pueden ver con un editor de textos.
Los ficheros binarios no.

La E/S en **ficheros de texto** se *debería* hacer con los operadores:
>> << (y a veces get(), getline() o put()).

La E/S en **ficheros binarios** se *debería* hacer con los operadores:
read() write() get() put()
No se deben usar << ni >>

Ventajas de los ficheros binarios:
- Ocupan menos *espacio* (normalmente).
- Las operaciones de E/S son más *eficientes* (operaciones en bloque).

Ventajas de los ficheros de texto:
- Se puede *ver y modificar* su contenido desde *cualquier editor*.

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 8

Metodología de la Programación

Tema 10.- Flujos asociados a ficheros (fstream)

- Introducción**
- Tipos de ficheros**
- Operaciones específicas**



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 9

Operaciones específicas con ficheros

Apertura y cierre

Operaciones **nuevas** sobre **fstream**, **ifstream** y **ofstream**:

Abrir un fichero para transferir datos.

Antes de poder enviar o recibir datos al fichero hemos de abrirlo.

Cerrar un fichero para finalizar la transferencia.

Cuando hemos acabado de trabajar con el fichero hemos de cerrarlo. Esta operación garantiza que el buffer del flujo se transfiere por completo y además se envía el carácter EOF si el fichero es de escritura.

```

    Abrir
    ↓
    Transferir
    ↓
    Cerrar
  
```



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 10

Operaciones específicas con ficheros

Apertura de un fichero



open()

```
void open(const char *nombrefich, openmode op);
          ^_____
          |       |
          |       optional
```

Asocia el fichero llamado *nombrefich* al flujo y lo abre. El modo de apertura (E/S) dependerá del tipo de flujo (*ifstream*, *ofstream* o *fstream*).

Podemos poner algunas banderas en el modo de apertura:

<code>ios::app</code>	Posicionamos el puntero al final (escribimos al final).
<code>ios::trunc</code>	Si el fichero existe lo borra y crea uno nuevo.

```
ifstream fi;
ofstream fo;
fi.open("ficherodeentrada.txt"); // Abrimos fichero de E/
fo.open("ficherodesalida.txt"); // Abrimos fichero de /S
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 11

Operaciones específicas con ficheros

Apertura de un fichero



Podemos usar un constructor sobrecargado para abrir el fichero cuando creamos el flujo:

```
ifstream fi("ficherodeentrada.txt"); // Abrimos fichero de E/
ofstream fo("ficherodesalida.txt"); // Abrimos fichero de /S
fstream fich("fich.ext"); // Abrimos/creamos fichero E/S
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 12

Operaciones específicas con ficheros

Cierre de un fichero



```

close()

void close();

Cerramos el fichero. Eliminamos la asociación que existe entre el
fichero y el flujo.

...
fi.close();
fo.close();
fich.close();

```

No se destruye el objeto flujo.

Podemos volver a asociar otro fichero al flujo.
close() es llamado por el destructor del flujo.

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 13

Operaciones específicas con ficheros

Ejemplo




```

#include <iostream>
#include <fstream>
using namespace std;
int main(int argc, char *argv[])
{
    ofstream f;
    f.open("mifichero.txt");
    f << 123 << endl;
    f.close();
}

```

Leemos un número
desde el fichero

```

#include <iostream>
#include <fstream>
using namespace std;
int main(int argc, char *argv[])
{
    ifstream f;
    int x;
    f.open("mifichero.txt");
    f >> x;
    f.close();
}

```

Escribimos 123 en el fichero

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 14

Operaciones específicas con ficheros

Comprobación de errores

Al abrir un fichero **conviene comprobar** si ha habido algún **error** (*el fichero no existe, el disco está lleno, no tenemos permisos, ...*)

```

ifstream f;
f.open("mifichero.txt");
if (f.fail()) {
    cerr << "Error al abrir el fichero" << endl;
    exit(1); // No podemos trabajar
}
...
f.close(); // Sólo podemos cerrar si ha sido abierto

```

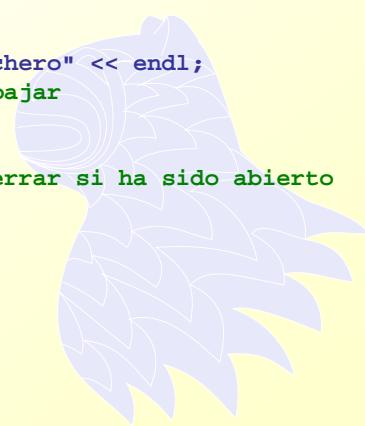
```

if (f.fail()) ...

también se puede poner como:

if (!f) ...

```



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 15

Operaciones específicas con ficheros

Generalidad de los flujos

Hemos visto que la funcionalidad de flujos y ficheros es la misma (ya que los ficheros se pueden ver como un caso particular de flujo)

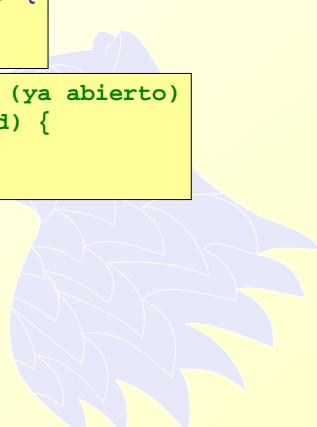
```

// Escribe una cadena en un flujo
void escribe(ostream &f, string cad) {
    f << cad;
}

// Escribe una cadena en un fichero (ya abierto)
void escribe(ofstream &f, string cad) {
    f << cad;
}

...
{
    string ca="hola";
    ofstream fich("fichero.txt");
    escribe(cout,ca);
    escribe(fich,ca);
    fich.close();
}

```



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 16

Operaciones específicas con ficheros

Generalidad de los flujos

La segunda versión NO es necesaria:
Un **ofstream** es un tipo particular de **ostream** y por tanto en cualquier sitio donde aparezca un **ostream** puede ir un **ofstream**.

```
// Escribe una cadena en un flujo  
void escribe(ostream &f, string cad) {  
    f << cad;  
  
// Escribe una cadena en un fichero (ya abierto)  
void escribe(ofstream &f, string cad) {  
    f << cad;  
}  
  
...  
{  
    string ca="hola";  
    ofstream fich("ficherito.txt");  
    escribe(cout,ca);  
    escribe(fich,ca);  
    fich.close();  
}
```

Lo mismo ocurre con:
ifstream ↔ istream
fstream ↔ iostream

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 17