Tema 3: Ficheros en C++

Programación 2

Curso 2013-2014

Notas

Tema 3

Ficheros de texto
Definición
Declaración
Apertura y cierre
Lectura
Escritura
Ejercicios
Ficheros binarios
Definición
Declaración, apertura y cierre
Lectura
Escritura
Ejercicios
Errores

Índice

Notas

¿Qué es un fichero de texto?

- También se le denomina *fichero con formato*
- Es un fichero que contiene solamente caracteres imprimibles
- ¿Qué es un carácter imprimible? Aquel cuyo código ASCII es mayor o igual que 32.
- ¿Qué es el código ASCII? Es un código que asigna a cada carácter un número (los ordenadores solamente almacenan números).
- Ejemplos de ficheros de texto: un programa en C++, una página web (HTML), un `makefile`, ...

Notas

Declaración de variables de tipo fichero

- Hay que poner `#include <fstream>` :
  - `ifstream fich_leer;` (sólo para leer)
  - `ofstream fich_escribir;` (sólo para escribir)
  - `fstream fich_leer_y_escribir;` (raro en ficheros de texto)

Notas

Apertura y cierre (1/2)

- Modos de apertura: lectura, escritura, lectura/escritura, añadir al final
- En C++ se abren los ficheros con "`open`", p.ej.:

```
const char nombre[]="mifichero.txt";
fichero.open(nombre,ios::in);
```

Si el nombre del fichero es un `string` debemos convertirlo con la función `c_str()`.

- Modos de apertura en C++:

lectura	`ios::in`
escritura	`ios::out`
lectura/escritura	`ios::in | ios::out` (`fstream`)
añadir al final	`ios::out | ios::app`

Notas

Apertura y cierre (2/2)

- Forma abreviada:

```
ifstream fl("ifi.txt"); // por defecto, ios::in

ofstream fe("ofi.txt"); // ios::out
```

- ¿Cómo comprobar si se ha abierto el fichero? ¿Cómo cerrarlo?

```
if (fichero.is_open())
{
   // ya se puede leer ...

   fichero.close();  // cerrar el fichero
}
else // error de apertura
```

Notas

Detección del fin de fichero

Se utiliza el método "`eof`":

```
ifstream fi;
...
  while (!fi.eof() ...)
```

¿Cómo funciona?

- Cuando se intenta leer un dato (carácter, número, etc) que ya no está en el fichero el método devuelve "`true`"
- OJO: después de haber leido el último dato válido sigue devolviendo "`false`", luego ...
- Es necesario hacer una lectura "extra" (que puede devolver datos no válidos que deben ignorarse) para provocar la detección del final del fichero

Notas

Lectura por líneas (1/2)

```
...
if (fi.is_open())
{
   getline(fi,s);  // 's' es de tipo string
   // fi.getline(cad,tCAD);  // 'cad' es 'char []'
   while (!fi.eof())
   {
      // hacer algo con 's'

      getline(fi,s);
   }

   fi.close();
}
```

Notas

Lectura por líneas (2/2)

¿Y si la última línea del fichero no tiene "\n"? se pierde y no se procesa!!

```
...
if (fi.is_open())
{
   s="";
   getline(fi,s);
   while (!fi.eof() || s.length()!=0)
   {
      // hacer algo con 's'

      s="";  // inicializar 's'
      getline(fi,s);
   }

   fi.close();
}
```

Notas

Lectura carácter a carácter

```
...
if (fi.is_open())
{
   c = fi.get();
   while (!fi.eof())
   {
      // hacer algo con 'c'

      c = fi.get();
   }

   fi.close();
}
```

Notas

Lectura de más de una variable

Los ficheros son "`stream`", funcionan como `cin` y `cout`

```
#include <fstream>
...
  ifstream fi;
  int numentero; double numreal;

  fi.open("mifichero.txt",ios::in);
  if (fi.is_open())
  {
     fi >> numentero;
     while (!fi.eof())
     {
         fi >> numreal;
         // hacer algo con 'numentero' y 'numreal'
         fi >> numentero; // lectura "extra" ?
     }
     fi.close();
  }
```

Notas

Escritura de un fichero de texto

```
  ofstream fo;

  fo.open("mifichero.txt",ios::out);
  if (fo.is_open())
  {
     fo << "Un numero entero: " << numentero << endl;
     ...

     fo.close();
  }
```

Notas

Ejercicios (1/5)

Ejercicio 1

Haz un programa que lea un fichero "`fichero.txt`" y escriba en otro fichero "`FICHERO2.TXT`" el contenido del fichero de entrada con todas las letras en mayúsculas.

Ejemplo:

`fichero.txt`	`FICHERO2.TXT`
`Hola, mundo.`	`HOLA, MUNDO.`
`Como estamos?`	`COMO ESTAMOS?`
`Adios, adios...`	`ADIOS, ADIOS...`

Notas

Ejercicios (2/5)

Ejercicio 2

Haz un programa que lea dos ficheros de texto, "`f1.txt`" y "`f2.txt`", y escriba por pantalla las líneas que sean distintas en cada fichero, con "`< `" delante si la línea corresponde a "`f1.txt`", y "` >`" si corresponde a "`f2.txt`".

Ejemplo:

`f1.txt`	`f2.txt`
`hola, mundo.`	`hola, mundo.`
`como estamos?`	`como vamos?`
`adios, adios...`	`adios, adios...`

La salida debe ser:

```
< como estamos?
> como vamos?
```

Notas

Ejercicios (3/5)

Ejercicio 3

Diseña una función "`finfichero`" que reciba dos parámetros: el primero debe ser un número entero positivo *n*, y el segundo el nombre de un fichero de texto. La función debe mostrar por pantalla las *n* últimas líneas del fichero.

Ejemplo:

```
$ finfichero(3,"cadenas.txt")

with several words
unapalabra
muuuuchas palabras, muchas, muchas...
```

Notas

Ejercicios (4/5)

Ejercicio 3 (sigue)

Hay dos soluciones:

1. A lo *"bestia"*: leer el fichero para contar las líneas que tiene, y volver a leer el fichero para escribir las *n* líneas finales
   PROBLEMA: ¿y si el fichero tiene 100000000000000 líneas?
2. Utilizar un vector de `string` de tamaño *n* que almacene en todo momento las *n* últimas líneas leídas (aunque al principio tendrá menos de *n* líneas)

Notas

Ejercicios (5/5)

Ejercicio 4

Dados dos ficheros de texto "`f1.txt`" y "`f2.txt`", en los que cada línea es una serie de números separados por "`:`", y suponiendo que las líneas están ordenadas por el primer número de menor a mayor en los dos ficheros, haz un programa que lea los dos ficheros línea por línea y escriba en el fichero "`f3.txt`" las líneas comunes a ambos ficheros, como en el siguiente ejemplo:

Ejemplo:

f1.txt	f2.txt	f3.txt
10:4543:23	10:334:110	10:4543:23:334:110
15:1:234:67	12:222:222	15:1:234:67:881:44
17:188:22	15:881:44	20:111:22:454:313
20:111:22	20:454:313	

Notas

¿Qué es un fichero binario?

- También se le denomina *fichero sin formato*
- Un fichero binario es un fichero en el que los datos se almacenan tal y como se almacenan en la memoria del ordenador, no se convierten en caracteres para almacenarlos.
- Normalmente, cada elemento del que se quiere guardar información se almacena en un *registro* (`struct`)
- Si los elementos se almacenan con registros, se puede acceder directamente al n-ésimo elemento sin leer los $n-1$ anteriores.
- Los ficheros de texto son ficheros de acceso secuencial, y los binarios son ficheros de acceso directo (o acceso aleatorio)

Notas

Declaración de variables, apertura y cierre de ficheros binarios

- Declaración de variables: como en los ficheros de texto:

```
ifstream fbl;   // fichero para lectura
ofstream fbe;   // fichero para escritura
```

- Apertura del fichero: hay que añadir "`ios::binary`"

```
fbl.open("mifichero.dat", ios::in | ios::binary);
fbe.open("mifichero.dat", ios::out | ios::binary);
```

- Cierre del fichero: como en los ficheros de texto, con `close`:

```
fbl.close();
```

- Otros modos de apertura:
  - lectura/escritura `ios::in | ios::out | ios::binary`
  - añadir al final `ios::out | ios::app | ios::binary`

Notas

Lectura de un fichero binario

```
typedef struct { ... } TIPOCIUDAD;
...

  TIPOCIUDAD ciudad;

  fbl.open("mifichero.dat",ios::in | ios::binary);
  if (fbl.is_open())
  {
     fbl.read((char *)&ciudad, sizeof(ciudad));
     while (!fbl.eof())
     {
        // procesar 'ciudad'

       fbl.read((char *)&ciudad, sizeof(ciudad));
     }
     fbl.close();
  }
```

Notas

Acceso directo a un elemento

La posición de un elemento en el fichero se puede calcular en función del tamaño (`sizeof`) de lo que hay antes.

```
if (fbl.is_open())
{
  // posicionar para leer el tercer elemento
  fbl.seekg ( (3-1)*sizeof(ciudad), ios::beg);
  fbl.read( (char *)&ciudad, sizeof(ciudad) );
  ...
}
```

Otras referencias para la posición:

`fbl.seekg( pos, ios::cur)` desde la posición actual
`fbl.seekg( pos, ios::end)` desde el final del fichero

OJO: `pos` puede ser negativo

Notas

Escritura de un fichero binario

```
typedef struct { ... } TIPOCIUDAD;
...

  TIPOCIUDAD ciudad;
  ofstream fbe;

  fbe.open("mifichero.dat",ios::out | ios::binary);
  if (fbe.is_open())
  {
     // rellenar 'ciudad'

     fbe.write((const char *)&ciudad, sizeof(ciudad));
     ...
  }
```

Notas

Acceso directo a un elemento para escribir

Hay que utilizar el método `seekp` (posicionamiento para escritura) en lugar de la `seekg` (para lectura).

```
if (fbe.is_open())
{
  // posicionar para escribir el tercer elemento
  fbe.seekp ( (3-1)*sizeof(ciudad), ios::beg);
  fbe.write( (const char *)&ciudad, sizeof(ciudad) );
  ...
}
```

ATENCIÓN: si la posición a la que se va a ir con `seekp` no existe en el fichero, se *alarga* el fichero para que se pueda escribir.

Notas

Escritura/lectura de registros con cadenas de caracteres

Si se desea almacenar un registro en un fichero binario que contenga una cadena de caracteres, se debe utilizar un vector de caracteres, nunca un `string`. Puede ser necesario *recortar* el string:

```
char cad[tcREG];
string s;

...
strncpy(cad,s.c_str(),tcREG-1);
cad[tcREG-1]='\0'; // strncpy no pone el \0 si no aparece
```

Notas

Información sobre la posición actual

- La posición actual del puntero de lectura (en bytes) se puede ver mediante la función `tellg`, y la de escritura mediante `tellp`. Ejemplo:

```
 // Colocamos el puntero de lectura al final
f.seekg(0,ios::end);

// Calculamos el numero de registros del fichero
cout << f.tellg()/sizeof(miRegistro) << endl;
```

Notas

Ejercicios

Ejercicio 5

Dado un fichero binario "`alumnos.dat`" que tiene registros de alumnos, con la siguiente información por cada alumno:

dni	vector de 10 caracteres
apellidos	vector de 40 caracteres
nombre	vector de 20 caracteres
turno	entero

Haz un programa que imprima por pantalla el DNI de todos los alumnos del turno 7.

Ampliación: haz un programa que intercambie los alumnos de los turnos 4 y 8 (los turnos van del 1 al 10).

Notas

Ejercicios

Ejercicio 6

Dado el fichero "`alumnos.dat`" del ejercicio anterior, haz un programa que pase a mayúsculas el nombre y los apellidos del quinto alumno del fichero, y lo vuelva a escribir en el fichero.

Ejercicio 7

Diseña un programa que construya el fichero "`alumnos.dat`" a partir de un fichero de texto "`alu.txt`" en el que cada dato (dni, nombre, etc) está en una línea distinta. Ten en cuenta que en el fichero de texto el dni, nombre y apellidos pueden ser más largos que los tamaños especificados para el fichero binario, en cuyo caso se deben recortar.

Notas

Tema 3

Ficheros de texto
Definición
Declaración
Apertura y cierre
Lectura
Escritura
Ejercicios
Ficheros binarios
Definición
Declaración, apertura y cierre
Lectura
Escritura
Ejercicios
Errores

Ejercicios

Ejercicio 8

Escribe un programa que se encarge de la asignación automática de alumnos en 10 turnos de prácticas. A cada alumno se le asignará el turno correspondiente al último número de su DNI (a los alumnos con DNI acabado en 0 se les asignará el turno 10). Los datos de los alumnos están en un fichero "`alumnos.dat`" con la misma estructura que en los ejercicios anteriores.
La asignación de turnos debe hacerse leyendo el fichero una sola vez, y sin almacenarlo en memoria. En cada paso se leerá la información correspondiente a un alumno, se calculará el turno que le corresponde, y se guardará el registro en la misma posición.

Notas

Errores de lectura/escritura (1/2)

- Se producen tanto en ficheros de texto como en ficheros binarios
- Es raro que se produzcan, pero hay que tenerlos previstos
- Es recomendable comprobar que no hay errores después de cada lectura/escritura
- Se debe utilizar el método `fail` (aunque hay otras formas):

```
if (fi.fail() && !fi.eof()) // error de lectura
     ...
```

Notas

Errores de lectura/escritura (2/2)

```
if (fi.is_open())
{
   bool error=false;
   getline(fi,s);
   if (fi.fail() && !fi.eof()) error=true;
   while (!error && !fi.eof())
   {
      // hacer algo con 's'

      getline(fi,s);
      if (fi.fail() && !fi.eof()) error=true;
   }

   if (error)
      // mensaje de error
   fi.close();
}
```

Notas