



Guion de prácticas 4

*Entrada y salida básica con
ficheros de texto*

Abril de 2015



Metodología de la Programación

Curso 2014/2015

Índice

1. Definición del problema	5
2. Objetivos	5
3. Manejo de ficheros de texto	5
3.1. Procedimiento para leer datos desde un fichero	5
3.2. Procedimiento para escribir datos en un fichero	7
4. Gestión básica de errores con ficheros	8
5. Manejar imágenes PGM de texto	9
6. Material a entregar	10

1. Definición del problema

Por ahora los únicos datos con los que pueden trabajar nuestros programas son los que introducimos desde el teclado o los que mostramos por la pantalla, pero en programas reales esto no es suficiente, sino que se hace necesario almacenar de forma permanente tanto los datos de entrada como los de salida de un mismo programa, para lo que se recurre al uso de ficheros en memoria masiva. En esta práctica introduciremos muy brevemente el protocolo de manejo de ficheros de texto, dejando los ficheros binarios y los detalles más avanzados del uso de ficheros para su introducción en temas posteriores.

2. Objetivos

- Conocer la estructura de los ficheros de texto.
- Leer datos de un fichero de texto.
- Escribir datos en un fichero de texto.

3. Manejo de ficheros de texto

Los ficheros de texto contienen datos que han sido codificados como texto usando un esquema como ASCII o UNICODE y, a diferencia de los ficheros binarios, aparecen exactamente como en la pantalla del ordenador, es decir, como una secuencia de caracteres. Por tanto un fichero de texto puede abrirse y editarse con programas editores como notepad en Windows o gedit en Ubuntu Linux.

3.1. Procedimiento para leer datos desde un fichero

Los datos de un fichero se pueden procesar de forma muy similar a como se leen datos desde el teclado o como se escriben datos en pantalla con los operadores de extracción `>>` y de inserción `<<`. En la Tabla 1 se pueden ver dos programas que leen exactamente los mismos valores para cada variable con la diferencia de que el primero los lee desde el teclado y el segundo los lee desde un fichero llamado "datos.txt".

Se puede ver que el procedimiento para leer los datos desde un fichero es muy sencillo y consiste en los siguientes pasos.

1. Incluir el fichero de cabeceras `fstream` para poder manejar ficheros.
2. Crear un flujo de datos de entrada `ifstream` que proviene de un fichero cuyo nombre es `fentrada` y asociarlo al fichero `datos.txt` del que se van a leer los datos.


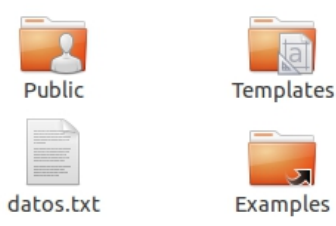
Código C++	Datos introducidos desde el teclado
<pre>#include <iostream> using namespace std; int main() { int i; double d; char c[64]; cin >> i; cin >> d; cin >> c; return 0; }</pre>	 <p>10 3.14 Pepe</p>
<pre>#include <iostream> #include <fstream> using namespace std; int main() { int i; double d; char c[64]; ifstream fentrada; fentrada.open("datos.txt"); fentrada >> i; fentrada >> d; fentrada >> c; fentrada.close(); return 0; }</pre>	 <p>10 3.14 Pepe</p>

Tabla 1: Dos programas y sus datos asociados, que leen exactamente los mismos valores para cada variable.


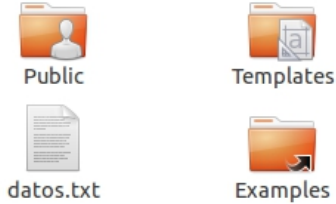
Código C++	Datos visualizados en pantalla
<pre>#include <iostream> using namespace std; int main() { int i=10; double d=3.14; char c[64]="Pepe"; cout << i << " "; cout << d << " "; cout << c << endl; return 0; }</pre>	 10 3.14 Pepe
Código C++	Datos guardados en el fichero datos.txt
<pre>#include <iostream> #include <fstream> using namespace std; int main() { int i=10; double d=3.14; char c[64]="Pepe"; ofstream fsalida; fsalida.open("datos.txt"); fsalida << i << " "; fsalida << d << " "; fsalida << c << endl; fsalida.close(); return 0; }</pre>	 10 3.14 Pepe

Tabla 2: Dos programas que escriben exactamente los mismos valores para cada variable en la pantalla (el primero) y en el fichero "datos.txt"(el segundo).

3. Leer los datos con el operador de extracción >> exactamente igual que si se leyese desde cin.
4. Cuando hemos terminado de leer los datos cerramos el flujo de entrada desde el fichero.

3.2. Procedimiento para escribir datos en un fichero

Para escribir datos en un fichero el procedimiento es muy similar y aparece ilustrado en los ejemplos de la Tabla 2.

En este caso, el procedimiento para guardar o escribir los datos en un fichero también es muy sencillo y consiste en los siguientes pasos.

1. Incluir el fichero de cabeceras `fstream` para poder manejar ficheros.
2. Crear un flujo de datos de salida `ofstream` hacia de un fichero cuyo nombre es `fsalida` y asociarlo al fichero `datos.txt` en el que se

van a escribir los datos. Si el fichero no existe, se crea, y si ya existe, se borran sus datos antes de empezar a escribir en él.

3. Escribir los datos en el fichero con el operador de inserción << exactamente igual que si se mostrasen por pantalla con cout.
4. Cuando hemos terminado de escribir los datos cerramos el flujo de salida.

4. Gestión básica de errores con ficheros

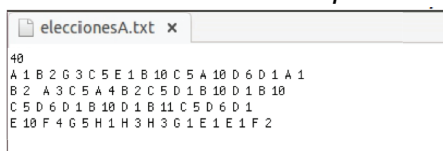
Son muchas las casuísticas que se pueden presentar cuando se manejan datos desde o hacia un fichero. En esta práctica introductoria se van a comprobar los posibles errores en la apertura y en el flujo de lectura / escritura de un fichero. El primer error se produce cuando se intentan leer datos de un fichero que no existe o cuando no se tienen privilegios para abrir el fichero o para escribir en él. El segundo error puede ocurrir cuando las lecturas / escrituras no se han producido con éxito. A continuación se muestra un ejemplo de la comprobación de errores asociados a las operaciones con ficheros:

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

int main(){
    int i;
    double d;
    char c[64];
    ifstream fentrada;
    fentrada.open( "datos.txt" );
    if (fentrada){
        fentrada >> i;
        fentrada >> d;
        fentrada >> c;
        if (!fentrada){
            cerr <<"error_de_lectura_del_fichero\n";
        }
        fentrada.close();
    }
    else{
        cerr << "error_de_apertura_del_fichero\n";
    }
}
```

Ejercicio 1 En un fichero de texto como el mostrado en la imagen se tienen recogidos los votos emitidos para un partido político (A, B, C, ..., H) desde distintas mesas indicando en la primera fila del fichero el número de

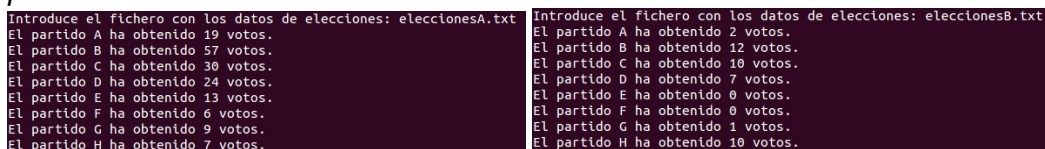
entradas que contiene el fichero. Cada entrada se compone de una letra ('A' - 'H') que representa al partido votado y de un número que indica el número de votos emitidos para ese partido.



```

48
A 1 B 2 G 3 C 5 E 1 B 10 C 5 A 10 D 6 D 1 A 1
B 2 A 3 C 5 A 4 B 2 C 5 D 1 B 10 D 1 B 10
C 5 D 6 D 1 B 10 D 1 B 11 C 5 D 6 D 1
E 10 F 4 G 5 H 1 H 3 H 3 G 1 E 1 E 1 F 2
  
```

Se pide construir un programa que lea desde el teclado el nombre del fichero de datos, abra el fichero, lea los datos, contabilice el número total de votos acumulado por cada partido, y muestre los resultados en orden alfabético por el nombre del partido. Las siguientes figuras muestran la salida de este programa después de procesar los ficheros `eleccionesA.txt` y `eleccionesB.txt` incluidos entre los ficheros de soporte de esta práctica:



Introduce el fichero con los datos de elecciones: eleccionesA.txt	Introduce el fichero con los datos de elecciones: eleccionesB.txt
El partido A ha obtenido 19 votos.	El partido A ha obtenido 2 votos.
El partido B ha obtenido 57 votos.	El partido B ha obtenido 12 votos.
El partido C ha obtenido 30 votos.	El partido C ha obtenido 10 votos.
El partido D ha obtenido 24 votos.	El partido D ha obtenido 7 votos.
El partido E ha obtenido 13 votos.	El partido E ha obtenido 0 votos.
El partido F ha obtenido 6 votos.	El partido F ha obtenido 0 votos.
El partido G ha obtenido 9 votos.	El partido G ha obtenido 1 votos.
El partido H ha obtenido 7 votos.	El partido H ha obtenido 10 votos.

Ejercicio 2 Ampliar el ejercicio 1 para guardar la salida en un fichero de texto en vez de mostrarla por pantalla.

5. Manejar imágenes PGM de texto

En esta última sección se van a utilizar imágenes PGM como las de la práctica 3 pero codificadas en texto. Las diferencias entre ambos tipos de imágenes son las siguientes:

- La primera línea del fichero es diferente (ver Figura 1), siendo P2 para las de texto y P5 para las binarias. A continuación vienen el número de columnas (256) y de filas (256), y el valor máximo de todos de grises que contiene (siempre 255, independientemente del valor máximo real).
- El contenido de la imagen de texto es perfectamente legible con un editor de texto mientras que la imagen binaria no lo es.

Ejercicio 3 Ampliar los ficheros `pgm.cpp` y `pgm.h` con dos funciones nuevas para leer imágenes PGM de texto basadas en las funciones para leer imágenes binarias y la descripción del formato PGM antes mencionada.

Ejercicio 4 Modificar el fichero `imagen.cpp` de la práctica 3 para que se puedan leer tanto imágenes de texto como binarias, con las funciones primitivas elaboradas en el Ejercicio 3, y para grabar las imágenes siempre en formato de texto.

Ejercicio 5 Recompilar los tres binarios de la práctica 3 (no es necesario modificar nada más) y comprobar que todo funciona correctamente.

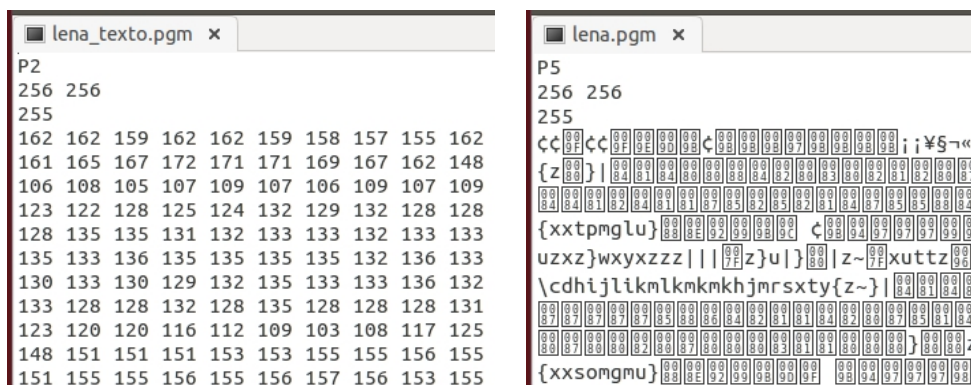


Figura 1: Contenido de la imagen 'lena.pgm' en formato de texto (izquierda) y binario (derecha).

6. Material a entregar

Cuando esté todo listo y probado el alumno empaquetará la estructura de directorios anterior en un archivo con el nombre **practica4.zip** y lo entregará en la plataforma **decsai** en el plazo indicado. No deben entregarse archivos objeto (.o) ni ejecutables. Para asegurarse de esto último conviene ejecutar **make mrproper** antes de proceder al empaquetado. Los ficheros deberán estar en los directorios adecuados: include para los ficheros .h y src para los ficheros .cpp.

El alumno debe asegurarse de que ejecutando las siguientes órdenes se compila y ejecuta correctamente su proyecto:

```
unzip practica4
cd practica4
make
bin/esteganografia
```

Se deberá incluir un informe en pdf donde aparezcan los integrantes de la pareja que hizo el trabajo, las dificultades que hayan tenido en realizar la práctica, y capturas de pantalla mostrando que los programas funcionan correctamente. Este informe se guardará en la carpeta doc.