LABORATORIO 10

FICHEROS EN EL LENGUAJE C

1 OBJETIVOS

Al finalizar esta actividad, el alumno será capaz de:

- Trabajar adecuadamente con ficheros.
- Utilizar funciones que manejen *ficheros.*
- Entender la diferencia entre los ficheros y las variables que los manipulan.
- Entender la diferencia entre los ficheros y los nombres de los ficheros.
- Utilizar los ficheros realizando acceso secuencial y directo.
- Comprobar la utilidad de la utilización de ficheros.

2 MOTIVACIÓN

- Existen multitud de problemas que no se pueden solucionar fácilmente si no es con el uso de ficheros.
- La utilización de ficheros simplifica la entrada y salida de datos, ya que almacenándolos una vez, se pueden utilizar todas las veces que deseemos.
- Con pequeñas variaciones los problemas que se han hecho hasta ahora se pueden utilizar con ficheros, permitiendo almacenar permanentemente los resultados obtenidos. Por ejemplo, si se obtiene un vector desde el teclado, se puede almacenar permanentemente en un fichero de disco.
- Además, los resultados obtenidos en un programa, si se almacenan en un fichero de disco, pueden reutilizarse en otros programas.
- Además, si se verifica que un dato es erróneo, puede modificarse sin necesidad de tener que modificar el resto de datos introducidos.
- Es posible editar directamente los datos y luego utilizarlos.
- Es posible generar automáticamente los datos, para luego utilizarlos.
- Es posible almacenar los resultados y luego comprobarlos adecuadamente.

3 USO DE FICHEROS EN C

Los prototipos de las funciones que se pueden utilizar en este laboratorio son los siguientes:

```
FILE * fopen(const char nombre[], const char apertura[]);
int fprintf(FILE * fichero, const char formato[], …);
int fputs(const char cadena[], FILE *fichero);
int fscanf(FILE * fichero, char cadena[], …);
char * fgets(char cadena[], int n, FILE *fichero);
int fclose(FILE *fichero);
int fseek(FILE *fichero, long pos, int modo);
int feof(FILE *fichero);
char * strerror(int error);
int rand(void);
void srand(unsigned int semilla);
```

4 EJERCICIOS

4.1 Ejercicio 1

4.1.1 Objetivo 1

Almacenar en un fichero de disco, los datos introducidos desde el teclado.

4.1.2 Enunciado 1

1. Escribir un programa que nos permita almacenar en el fichero "float.txt" los datos introducidos por el teclado. El programa debe finalizar cuando se introduzca un número negativo.
2. Ejecutar el programa.
3. Comprobar que los datos introducidos por teclado se corresponden con los datos almacenados en el fichero.
4. Ejecutar el programa por segunda vez y comprobar los resultados.
5. Modificar el programa para que los datos introducidos se añadan al final del fichero.

Ejemplo de ejecución:
(los datos tecleados por el usuario están en *cursiva y subrayados*)

```
INTRODUCIR VALORES REALES POSITIVOS
X(1): 13.91
X(2): 23.11
X(3): 83.13
X(4): 12.22
X(1): -1
Presione una tecla para continuar . . .
```

En este caso la pantalla de ejecución del programa no tiene demasiado interés ya que los resultados se almacenarán en el fichero "float.txt". Se debe comprobar que el fichero contiene los datos que acabamos de introducir.

4.1.3 Ayuda para el enunciado 1

Escribir un programa cíclico que permita introducir datos desde el teclado, para luego almacenar dichos datos en un nuevo fichero.

Lo primero que se debe hacer es escribir un breve programa de introducción de datos. Es posible utilizar cualquier programa de los que se han escrito hasta ahora que solicite datos numéricos del usuario y finalice al introducir un número negativo.

Después de ejecutar el programa, los datos introducidos deberán estar almacenados en el fichero "float.txt". Eso se puede comprobar abriendo el fichero desde el entorno de desarrollo o bien desde el explorador de Windows.

Recordar que para almacenar los datos en un fichero, se puede utilizar la función:

int **fprintf**(FILE * fichero, const char formato[], …);

pero para ello es necesario disponer con anterioridad de una variable fichero, a la cual se le puede asignar valor al utilizar la función:

FILE * **fopen**(const char nombre[], const char apertura[]);

Recordar que, en este caso, el modo de apertura debería ser “w” o “a”

Observar cómo afecta la modificación del modo de apertura al fichero que almacenará los resultados. Dependiendo del tipo de problema interesará un comportamiento u otro.

Observar que los datos se pueden almacenar en el fichero de formas diferentes dependiendo de la cadena de formato que se haya utilizado.

4.2 Ejercicio 2

4.2.1 Objetivo 2

Almacenar en disco datos generados por el programa.
Observar que existen algunos caracteres que no tienen representación visible.

4.2.2 Enunciado 2

1. Escribir un programa que almacene en el fichero “ascii.txt” el juego de caracteres ASCII.
2. Modificarlo para almacenar el “juego de caracteres extendido”

4.2.3 Ayuda para el enunciado 2

Recordar que algunos de los caracteres, aquellos cuyo código es menor que 32, no son representables gráficamente en la pantalla (por ejemplo el carácter 8 no se visualiza y suena un pitido)

El fichero debería tener un aspecto similar al siguiente:

….
65 A
66 B
67 C
68 D
69 E
76 L
…

En donde la primera columna representa los valores numéricos correspondientes a los caracteres de la segunda columna.

4.3 Ejercicio 3

4.3.1 Objetivo 3

Dado que ya se conoce cómo almacenar datos en los ficheros, ahora hay que utilizarlos, es decir, se usarán los datos numéricos que se encuentran en un fichero para obtener su suma y su producto (en general cualquier tipo de operaciones)

4.3.2 Enunciado 3

1. Escribir un programa que utilizando los datos del fichero "float.txt" obtenga su suma y su producto. Para ello definir dos funciones una para la suma y otra para el producto, con prototipos similares a los siguientes:

   float **sum_fich** (char * nomfich);
   float **mult_fich** (char * nomfich);

2. Comprobar que se pueden eliminar los decimales en el fichero "float.txt" y el programa funciona correctamente; es decir, es posible introducir los datos tanto con representación entera como con representación decimal.

3. Generalizar el programa anterior para recibir desde el teclado el nombre del fichero que se desea utilizar.

4.3.3 Ayuda para el enunciado 3

Utilizar el fichero "float.txt" obtenido anteriormente para editar o añadir nuevos valores.

4.4 Ejercicio 4

4.4.1 Objetivo 4

Comprobar la utilidad de la generación de números al azar para obtener datos almacenados en ficheros que serán utilizados posteriormente para la realización de test de programas.

4.4.2 Enunciado 4

1. Escribir una función que almacene en un fichero, cuyo nombre ha sido proporcionado por el usuario, una secuencia 100 de números generados al azar mediante la función **rand**().
2. Utilizando los datos del fichero, escribir una función que proporcione el valor medio, la varianza y la desviación típica.
3. Construir un fichero denominado "semilla.txt" que contenga un valor entero.
4. Modificar la función del punto primero con el fin de almacenar diferentes secuencias de números, usando el valor de la semilla almacenado en "semilla.txt"

4.4.3 Ayuda para el enunciado 4

- El valor de la semilla se puede almacenar directamente en un fichero utilizando el entorno de desarrollo para editar directamente el fichero "semilla.txt", o bien utilizar el programa para solicitar una nueva semilla al usuario.

4.5 Ejercicio 5

4.5.1 Objetivo 5

Escribir funciones que basadas en la función **rand**() permitan generar diferentes distribuciones aleatorias y utilizar ficheros mediante acceso directo.

4.5.2 Enunciado 5

1. Escribir una función que almacene en un fichero, denominado "sensores.txt", una tabla que represente a los sensores de una máquina, de un modo similar al siguiente:

```
0001000000
1000000000
0110000000
0000000A00
1000000000
0000000000
0000010000
0100001100
0000000001
0000000000
```

   La tabla anterior representaría una máquina con 100 sensores (10 filas y 10 columnas), en la que los estados de los sensores se representan mediante ceros y unos. El cero representa que el sensor está desactivado y el uno representa que el sensor está activado. Para representar un sensor en avería utilizamos la letra **A**.

   Para generar los valores de los estados de los sensores se ha de tener en cuenta que las **averías** se producen con una probabilidad de un 1% los sensores se encuentran **activados** con una probabilidad de un 10% y en **reposo** el resto de los casos.

   Se supone que en número de filas y el número de columnas son constantes conocidas; por ejemplo NF=10 y NC =10.

2. Contar el número de sensores averiados.
3. Abrir el fichero en modo directo y obtener el valor de la fila 4 y la columna 8.
4. Generalizar el punto anterior para obtener el valor correspondiente a una fila y a una columna cualquiera que se introduzcan por teclado hasta que se introduzca una fila o una columna negativa.
5. Utilizando un fichero denominado "serie.txt" que contiene en cada línea el nuevo valor del sensor y la fila y la columna en la que se encuentra dicho sensor, actualizar el fichero "sensores.txt" de forma automática.

4.5.3 Ayuda para el enunciado 5

- Escribir una función que genere el estado de un sensor.
- Para generar un número entre 1 y 100 puede utilizarse: rand()%100+1
- Tener en cuenta que al final de cada línea existen dos caracteres invisibles que representan el fin de línea. Esto es así, aunque se pueda pensar que sólo existe un carácter de fin de línea.