TEMA 4 - FICHEROS

1.- Apertura de ficheros

C nos permite el acceso a ficheros gracias a la biblioteca studio.h, que nos facilita todo lo necesario para el trabajo con ficheros. El acceso a un fichero puede ser:

a) Acceso secuencial: orientado a su uso en archivos donde la información está dispuesta como tipo texto.

b) Acceso aleatorio: orientado a su uso en archivos con datos contenidos en registros de longitud fija, que a su vez pueden estar subdivididos en campos. Un campo puede contener un valor numérico o ser tipo texto.

c) Acceso binario: permite guardar datos con el orden y estructura que se desee, siendo posible acceder a ellos conociendo la posición (número de byte) en que se encuentran.

Al existir distintas formas de acceso, se habla de "archivos secuenciales", "archivos aleatorios" y "archivos binarios", aunque esta terminología debe tomarse con precaución. Es decir, un "archivo binario" hace referencia a un archivo donde habitualmente guardamos y extraemos datos usando el acceso binario. Sin embargo, podríamos usar este archivo con otro tipo de acceso. En resumen, un archivo es "información" y la información puede reordenarse, cambiar, manipularse de distintas maneras, etc. Usar un tipo de acceso en un momento dado no obliga a usarlo siempre, aunque así pueda ser.

En la analogía fichero físico - fichero informático, las equivalencias aproximadas serían las de:

- Archivo secuencial = La información no está en campos de longitud predefinida.

- Archivo aleatorio = Existen campos de longitud predefinida, que pueden carecer de información por no estar disponible.

- Archivo binario = La información no está en campos de longitud predefinida pero se dispone de información referente a su ubicación.

Los ficheros, en contraposición con las estructuras de datos vistas hasta ahora (variables simples, vectores, registros, etc.), son estructuras de datos almacenadas en memoria secundaria. Para utilizar la información en memoria

principal se emplea fundamentalmente la instrucción de asignación; sin embargo, para guardar o recuperar información de un fichero es necesario realizar una serie de operaciones que describiremos en este apartado. El formato de declaración de un fichero es el siguiente:

FILE * nom_ var_fich;

En otros lenguajes la declaración del fichero determina el tipo de datos que se van a almacenar en él. En C la filosofía es distinta, todos los ficheros almacenan bytes y es cuando se realiza la apertura y la escritura cuando se decide cómo y qué se almacena en el mismo; durante la declaración del fichero no se hace ninguna distinción sobre el tipo del mismo. En la operación de apertura se puede decidir si el fichero va a ser de texto o binario, los primeros sirven para almacenar caracteres, los segundos para almacenar cualquier tipo de dato. Si deseamos leer un fichero como el autoexec.bat utilizaremos un fichero de texto, si queremos leer y escribir registros (struct) usaremos un fichero binario.

Antes de usar un fichero es necesario realizar una operación de apertura del mismo; posteriormente, si se desea almacenar datos en él hay que realizar una operación de escritura y si se quiere obtener datos de él es necesario hacer una operación de lectura. Cuando ya no se quiera utilizar el fichero se realiza una operación de cierre del mismo para liberar parte de la memoria principal que pueda estar ocupando (aunque el fichero en sí está almacenado en memoria secundaria, mientras está abierto ocupa también memoria principal)

La instrucción más habitual para abrir un fichero es :

FILE * fichero;
fichero = fopen (nombre-fichero, modo);

La función fopen devuelve un puntero a un fichero que se asigna a una variable de tipo fichero. Si existe algún tipo de error al realizar la operación, por ejemplo, porque se desee abrir para leerlo y éste no exista, devuelve el valor NULL.

El nombre-fichero será una cadena de caracteres que contenga el nombre (y en su caso la ruta de acceso) del fichero tal y como aparece para el sistema operativo. El modo es una cadena de caracteres que indica el tipo del fichero (texto o binario) y el uso que se va a hacer de él: lectura, escritura, añadir datos al final, etc. Está formado, por tanto, por dos componentes.

El primer componente puede ir acompañado de un símbolo +, por ejemplo r+. El símbolo + indica que se permite la lectura y la escritura. Por ejemplo r+ indica apertura de un archivo existente para lectura o escritura, w+ indica creación de un nuevo archivo para lectura o escritura y a+ indica apertura de un archivo para lectura o escritura al final.

El segundo componente indica el tipo de archivo que se abre.

Los modos disponibles son

Primer componente (modo)	Significado
r	Abre un archivo existente para lectura.
w	Crea un nuevo archivo para escritura (si ya existe, se pierden los datos preexistentes).
a	Abre el archivo para añadir datos al final (se conservan datos existentes), o crea el archivo si no existía.

Y los tipos

Segundo componente (tipo de archivo)	Significado
t	Archivo de texto.
b	Archivo binario.

La forma habitual de utilizar la instrucción fopen es dentro de una sentencia condicional que permita conocer si se ha producido o no error en la apertura, por ejemplo:

```
FILE *fich;
if ((fich = fopen("nomfich.dat", "r")) == NULL) {
 /* control del error de apertura * /
printf ( " Error en la apertura. Es posible que el fichero no   exista \n ");}
```

Cuando se termine el tratamiento del fichero hay que cerrarlo; si la apertura se hizo con fopen el cierre se hará con fclose (fich).

Para utilizar las instrucciones de manejo de ficheros es necesario incluir la librería <stdio.h>.

```
#include <stdio.h>
int main()
{
FILE *f = fopen("ejemplo.c", "r");
if (f==NULL)
{
   perror ("Error al abrir fichero.txt");
   return -1;
}
return 0;
}
```

2.- Lectura de ficheros

Para la recuperación de datos desde archivos tenemos las siguientes funciones disponibles.

Función o macro	Significado y ejemplo aprenderaprogramar.com
fgetc	Recupera un carácter del archivo fgetc (nombreInternoFichero);
getc	Igual que fgetc
fgets (arg1, arg2, arg3)	Recupera el contenido del archivo en la variable arg1 hasta que encuentra o bien un carácter de fin de línea (\n) o bien hasta extraer arg2-1 caracteres siendo arg2 un valor especificado en la llamada de la función y arg3 el nombre interno del fichero. Ejemplo: fgets (cadena1, n, nombreInternoFichero);
fscanf (arg1, "%format1 %format2 ...", &var1, &var2 ...)	Recupera el contenido del archivo con nombre interno arg1 (hasta el primer espacio en blanco) en las variables var1, var2 ... con los formatos %format1, %format2... Ejemplo: fscanf (fichero, "%c", &cadena1[i]); Ejemplo: fscanf (fichero, "%d %d", &numero1, &numero2);

Una de las posibles funciones que tenemos para leer un fichero de texto es fgets(). Esta función lee una línea completa del fichero de texto y nos la devuelve. Tiene los siguientes parámetros:

- char *s: Un array de caracteres en el que se meterá la línea leída del fichero. Este array debe tener tamaño suficiente para la línea o tendremos problemas.
- int n: Tamaño del array de caracteres. Si la línea es más larga, sólo se leeran n caracteres, para no desbordar el array. Si la línea tiene menos caracteres, se meteran en el array y se pondra un fin de cadena (un \0) al final.
- FILE* stream. Lo que obtuvimos con el fopen() del fichero que queremos leer.

Indicar que con fgets extraemos una línea completa hasta el salto de línea (siendo éste incluido dentro de la cadena extraída).

Esta función devuelve la misma cadena de caracteres que le pasamos si lee algo del fichero o NULL si hay algún problema (fin de fichero, por ejemplo).

Para leer todas las líneas consecutivamente del fichero, podemos hacer un bucle hasta que esa función fgets() nos devuelva NULL. Puede ser así

Ejercicio 1.

```
#include <stdio.h>
int main()
{
FILE *f = fopen("ejemplo.c", "r");
if (f==NULL)
{
  perror ("Error al abrir fichero.txt");
  return -1;
}
char cadena[100];
while (fgets(cadena, 100, f) != NULL)
{
 printf("%s \n",cadena);
}
```

```
return 0;
}
```

La sentencia fscanf extrae un dato (numérico o cadena de texto) hasta que encuentra un espacio, omitiendo el espacio. Puede resultar útil para extraer datos en un fichero delimitados por espacios, pero conviene tener en cuenta esta circunstancia si pretendemos extraer texto que contiene espacios, ya que únicamente nos extraerá la primera palabra o cadena, hasta que se presente un espacio. También sería posible especificar longitudes exactas a ocupar con fprintf y a extraer con fscanf.

Ejercicio 2. Lectura de un fichero con estructura.

Supongamos que un fichero está formado por datos agrupados en una estructura del siguiente tipo

Nombre
Domicilio
Edad.

Y que contiene los siguientes datos

Antonio pérez
calle la union, 27
34

Para leer un elemento de la estructura necesitaríamos el siguiente código

```
#include <stdio.h>
int main()
{
char nombre[50];
char dom[50];
int edad;
FILE *f = fopen("datos.txt", "r");
if (f==NULL)
{
   perror ("Error al abrir fichero.txt");
   return -1;
}

fgets(nombre, 51, f) ;
fgets(dom, 51, f) ;
fscanf(f,"%d",&edad);
printf("%s%s%d \n",nombre,dom,edad);
return 0;
}
```

Ejercicio 3
Si el fichero contiene varios registros con esa estructura tendríamos que hacer un bucle

```
#include <stdio.h>
int main()
{
char nombre[50];
char dom[50];
int edad;
FILE *f = fopen("datos.txt", "r");
if (f==NULL)
{
   perror ("Error al abrir fichero.txt");
   return -1;
}

while(fgets(nombre, 51, f)!=NULL) {
fgets(dom, 51, f) ;
fscanf(f,"%d",&edad);
printf("%s%s%d \n",nombre,dom,edad);
}
return 0;
}
```

Ejercicio 4

Si los datos del fichero están organizados en torno a esa estructura puede resultar más lógico definir la estructura y leerla del fichero:

#include <stdio.h>

```
int main()
{
struct ficha
  {
      char nombre[50];
      char dom[50];
      int edad;
  } datos[10];
  int i=0;

FILE *f = fopen("datos.txt", "r");
if (f==NULL)
{
   perror ("Error al abrir fichero.txt");
   return -1;
}
while(fgets(datos[i].nombre, 51, f)!=NULL) {
fgets(datos[i].dom, 51, f) ;
fscanf(f,"%d",&datos[i].edad);
printf("%s%s%d \n",datos[i].nombre,datos[i].dom,datos[i].edad);
i++;
}
return 0;}
```

Ejercicio 5

En determinadas circunstancias conviene cargar todos los datos del fichero en un array con esa estructura e imprimirlos despues.

```
#include <stdio.h>
int main()
{
            struct ficha
  {
     char nombre[50];
     char dom[50];
     int edad;
  } datos[10];
  int i=0;

FILE *f = fopen("datos.txt", "r");
if (f==NULL)
{
   perror ("Error al abrir fichero.txt");
   return -1;
}
while(fgets(datos[i].nombre, 51, f)!=NULL) {
fgets(datos[i].dom, 51, f) ;
fscanf(f,"%d",&datos[i].edad);
i++;
}
int j;
for (j=0;j<i;j++)
printf("%s%s%d \n",datos[j].nombre,datos[j].dom,datos[j].edad);
return 0;
}
```

3.-Escritura en Ficheros

Para escribir en ficheros utilizamos la función fprintf cuya sintaxis es similar a prinft pero añadiendo el puntero al fichero

fprintf(puntero,formato,variable)

En el siguiente ejemplo leemos datos desde teclado y los guardamos en un fichero hasta que el usuario introduzca un numero distinto de 0.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
FILE *f = fopen("datos1.txt", "w");
if (f==NULL)
{
   perror ("Error al abrir fichero.txt");
   return -1;
}
int con=0;
char usu[20];
while(con==0) {
    printf("Usuario: ");
```

```
    gets(usu);
    fprintf(f,"%s\n",usu);
    printf("Introducir más datos ");
    scanf("%d", &con);
    fflush(stdin);
}
fclose(f);
return 0;}
```

Ahora leemos los datos introducidos

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
FILE *f = fopen("datos1.txt", "r");
if (f==NULL)
{
   perror ("Error al abrir fichero.txt");
   return -1;
}
char nombre[20];
while(fgets(nombre, 51, f)!=NULL) {
     printf("%s",nombre);
}
return 0;}
```

En el siguiente ejemplo guardamos datos de tipo cadena mezclados con datos numéricos.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
FILE *f = fopen("datos1.txt", "w");
if (f==NULL)
{
   perror ("Error al abrir fichero.txt");
   return -1;
}
int i,edad=0;
char usu[20];
for(i=0;i<3;i++){
    printf("Usuario: ");
    gets(usu);
    fprintf(f,"%s\n",usu);
    printf("\nEdad ");
    scanf("%d", &edad);
     fprintf(f,"%d\n",edad);
    fflush(stdin);
}
fclose(f);
return 0;}
```