Tarea #4

Inyerman Alexander Xap Chin 20210097

Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala.

A. Resumen

En este trabajo investigamos y posteriormente generamos un programa que es capaz de recibir información dada por el usuario para guardarla en un archivo de texto. Para crear de esta manera una base de datos rudimentaria.

Obtuvimos el código del ingeniero el cual procedimos a recrear y probar para ver su funcionamiento en el cual nos percatamos de que no estaba guardando correctamente nuestros datos luego de descubrir el error en el código pudimos resolverlo y ver que el código corría como debía.

B. Objetivos

- Investigar como se crean y utilizan los ficheros desde un programa en lenguaje c.
- Crear un programa que guarde los datos dados por el usuario en un fichero.
- Utilizar la función Sleep en el programa

C. Marco Teórico

ESCRIBIR EN ARCHIVOS EN LENGUAJE C:

Ya tenemos definido cómo abrimos y cerramos la comunicación con un archivo. Ahora tenemos que ver cómo manipulamos los datos. Para ello nos valemos de las funciones disponibles. En el caso de escritura en archivos tenemos las siguientes.

fputc

Escribe un carácter en el archivo

Ejemplo:

fputc: (‘a’, nombreInternoFichero);

putc: Igual que fputc

fputs: Escribe una cadena de texto en el archivo

Ejemplo:

Fputs:

(“cadena”,nombreInternoFichero);

fprintf

Escribe como texto a un archivo los datos transformando el formato especificado en texto

Ejemplo: fprintf (fichero, "%s %d", cadena1, num);

LEER Y ESCRIBIR ARCHIVOS USANDO LENGUAJE C

Advertencia: una manipulación incorrecta de ficheros puede dar lugar a la pérdida de datos, al malfuncionamiento de programas o a fallos del sistema operativo del ordenador. Es necesario disponer de copias de seguridad de toda información que se considere importante.

Guardar datos a un disco o recuperar los datos previamente guardados son procesos fundamentales en cualquier programa informático. La

importancia del almacenamiento de contenidos es obvia: envío de información a otros usuarios, posponer el trabajo varios días o semanas sin tener que introducir manualmente los datos de nuevo, acceso a información almacenada en sistemas remotos, etc. Incluso para desarrollos de software de relativamente corta longitud resulta relevante la gestión de datos, por ahorrar una cantidad de tiempo considerable.

Ficheros y bases de datos constituyen un cuerpo de conocimiento de cierta complejidad y extensión. Por motivos de espacio no vamos a hacer más que una breve introducción.

Podemos pensar en los ficheros o archivos de forma análoga a lo que sería un archivo físico: un lugar donde hay información almacenada. Ahora bien, eso no nos dice cómo se encuentra esa información: si está ordenada (alfabéticamente, de mayor a menor, en impresos con campos predefinidos), si son datos de un tipo o de varios tipos (numéricos, tipo texto, mezcla de tipos de datos...), si se leen usando alguna clave de interpretación (una partitura no se lee igual que un texto), etc. Todo esto es relevante y por ello normalmente sabremos qué tipo de datos contiene el fichero y cómo están ordenados antes de acceder a él. No es lo mismo guardar "3246" como número tipo int que como cadena de caracteres. Un tipo int ocupa un espacio en disco, p.ej. 2 bytes y tiene una "clave" para su lectura, mientras que como caracteres cada cifra ocupa un byte y se lee como si de texto se tratara.

Un asunto para considerar es si la información está contenida en campos de longitud predefinida. Consideremos un archivo donde se almacenan los nombres, números de DNI y año de nacimiento de 3 personas. Podemos considerar diferentes casos:

ACCESO A FICHEROS CON C: SECUENCIAL, ALEATORIO O BINARIO.

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.

Repetimos que todo es lo suficientemente complejo como para que sea difícil dar definiciones exactas. Nosotros vamos a estudiar únicamente ciertas formas de extraer y guardar datos de un fichero secuencial a través de algunas funciones disponibles en el lenguaje para este fin. Esto será suficiente para poder crear programas que leen y guardan datos desde archivos de forma simple. El resto de las instrucciones y posibilidades de acceso no será objeto de estudio ya que quedan fuera de nuestros objetivos.

ABRIR Y CERRAR FICHEROS SECUENCIALES EN C. FILE, FOPEN, FCLOSE.

Para manipular información de un fichero, por ejemplo, guardar datos en él o leer datos desde él, lo primero que hemos de hacer es abrir el fichero. El proceso que seguir es:

Abrir el fichero --> Manipular datos (extraer, guardar, modificar, etc.) -->Cerrar el fichero

fopen --> Manipular datos --> fclose

A su vez, cada vez que accedemos a un fichero hemos de indicar “bajo qué modalidad” vamos a utilizar ese fichero: si es para leer datos, para escribir datos, y a su vez si se trata de escribir si debemos añadir los datos a los que ya existieran en el fichero o si debemos desechar lo existente y crear el fichero como si fuera nuevo.

La sintaxis que emplearemos para escribir o leer datos en ficheros secuenciales será la siguiente:

FILE* nombreInternoFichero;

nombreInternofichero = fopen (“rutaNombreFichero.txt”, “modoDeAcceso”);operación1;operación 2; … ; operación n;…

fclose (nombreInternofichero)

FILE* nombreInternoFichero: para manipular un fichero necesitamos “algo” que el programa utilizará para almacenar información sobre el fichero (como el modo de acceso,

ruta, etc.). Ese “algo” (estructura de datos) lo creamos escribiendo FILE* y el nombre que queramos utilizar para referirnos a la estructura de datos que almacenará la información relacionada con el fichero. Cuando creamos esta estructura de datos (en realidad deberíamos usar el término puntero, pero vamos a mantener las ideas claras usando simplificaciones) la creamos vacía (su contenido es NULL), a la espera de rellenarla con información.

nombreInternoFichero = fopen (“nombreFichero.txt”, “modoDeAcceso”): es la instrucción que da lugar a que la estructura de datos deje de estar vacía y comience a contener información y a ser útil para poder actuar sobre un fichero. Para aportar la información necesaria se usa la función fopen, que abre el fichero cuya ruta y nombre se especifica en rutaNombreFichero.txt (no necesariamente tiene que ser un fichero txt, pero de momento vamos a suponer que es así) y define el modo de acceso. El modo de acceso indica para qué se abre y cómo debe tratarse el fichero según si existía previamente o si no existía.

Operación 1; Operación 2; …: indica las operaciones que se ejecutan. Pueden ser bucles, condicionales, o instrucciones simples que normalmente se encargarán de extraer información del fichero (leer datos) o incorporar información al fichero (escribir datos).

fclose (nombreInternoFichero): da lugar a la conclusión de procesos abiertos, al cierre del fichero y a que “la línea de comunicación” quede libre para una posterior ocasión. Siempre que abramos un fichero con fopen deberemos cerrarlo con fclose.

MODOS DE ACCESO O APERTURA DE ARCHIVOS CON C. LEER (READ), ESCRIBIR (WRITE) O AÑADIR (APPEND).

Para acceder a un fichero se escriben sentencias similares a estas:

FILE* fichero; fichero = fopen("cursoAF1.txt", "wt");

wt es una clave identificadora que consta de dos componentes: w y t. Vamos a definir cuál es el significado de las dos componentes de “wt” y qué valores pueden aparecer en este lugar y su significado. La comprensión se facilita si tenemos en cuenta que r proviene de read, w proviene de write y a proviene de append.

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.

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.

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

D. Marco Practico

Se procede a recrear el código provisto por el ingeniero y a comprobar su funcionamiento luego nos percatamos de que no estaba escribiendo los datos en el archivo de texto procedimos a realizar nuestra investigación de escribir y leer datos de ficheros que esta reflejada en el marco teórico de este trabajo y nos percatamos de que la segunda variable no está declarada correctamente en:

FILE*archivo=fopen("salida.txt", "a");

Procedemos a decirle al programa que es un archivo de texto y darle permisos de escritura por lo que queda de la siguiente manera:

FILE*archivo=fopen("salida.txt", "a+t");

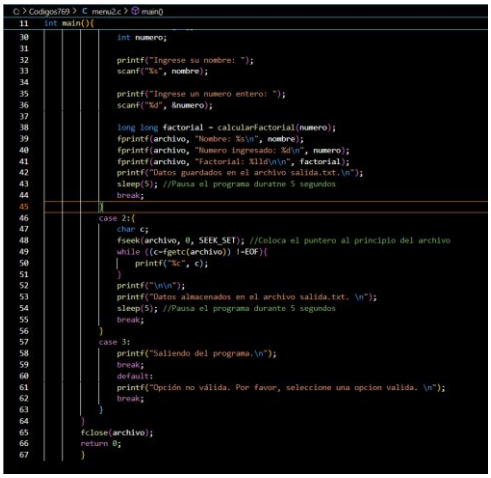
Procedemos a comprobarlo y el programa corre como se esperaba.

E. Código

Código en c parte 1/2

```c
#include <stdio.h>
#include <stdlib.h> //libreria para ejecutar instrucciones del SO
#include <unistd.h> //Inluye la biblioteca para la funcion sleep

long long calcularFactorial(int num) {
    if (num<=1){
        return 1;
    }
    return num*calcularFactorial(num - 1);
}
int main(){
    FILE*archivo=fopen("salida.txt", "a+t");
    system("clear");
    if (archivo == NULL){
        printf("No se pudo abrir el archivo.\n");
        return 1;
    }
    int opcion = 0;
    while  (opcion !=3) {
        system ("clear");
        printf("Menu: \n");
        printf("1. Ingresar datos \n");
        printf("2. Mostrar Historial \n");
        printf("3. Salir \n");
        printf("Seleccione una opcion: \n");
        scanf("%d", &opcion);
        switch (opcion) {
            case 1:{
                char nombre [50];
                int numero;
```

Código en c parte 2/2



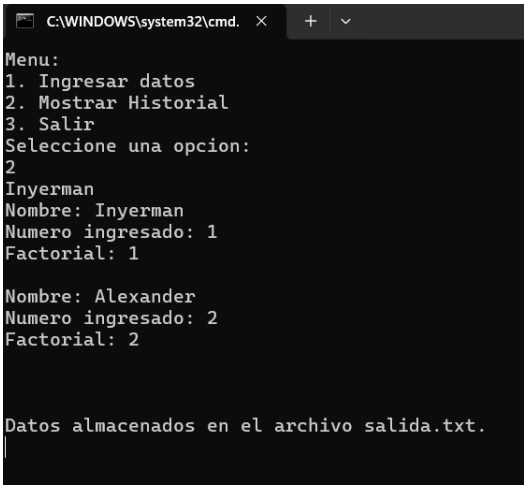
```c
                printf("Ingrese su nombre: ");
                scanf("%s", nombre);

                printf("Ingrese un numero entero: ");
                scanf("%d", &numero);

                long long factorial = calcularFactorial(numero);
                fprintf(archivo, "Nombre: %s\n", nombre);
                fprintf(archivo, "Numero ingresado: %d\n", numero);
                fprintf(archivo, "Factorial: %lld\n\n", factorial);
                printf("Datos guardados en el archivo salida.txt.\n");
                sleep(5); //Pausa el programa duratne 5 segundos
                break;
            }
            case 2:{
                char c;
                fseek(archivo, 0, SEEK_SET); //Coloca el puntero al principio del archivo
                while ((c=fgetc(archivo)) !=EOF){
                    printf("%c", c);
                }
                printf("\n\n");
                printf("Datos almacenados en el archivo salida.txt. \n");
                sleep(5); //Pausa el programa durante 5 segundos
                break;
            }
            case 3:
                printf("Saliendo del programa.\n");
                break;
                default:
                printf("Opción no válida. Por favor, seleccione una opcion valida. \n");
                break;
            }
        }
        fclose(archivo);
        return 0;
    }
```

F. Resultados

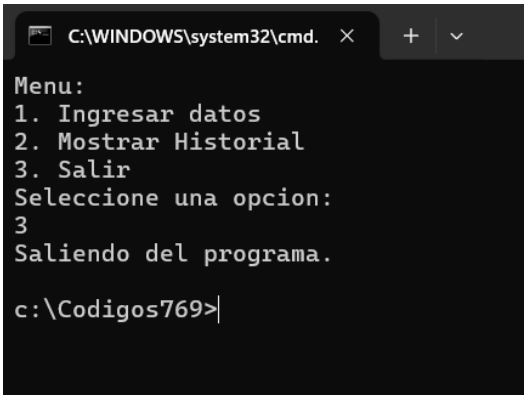
Código corriendo opción 1

```
Menu:
1. Ingresar datos
2. Mostrar Historial
3. Salir
Seleccione una opcion:
1
Ingrese su nombre: Inyerman
Ingrese un numero entero: 202100297
```

Código corriendo opción 2



Código corriendo opción 3



G. Conclusiones

- Investigamos como se crean y utilizan los ficheros desde un programa en lenguaje c.
- Creamos un programa que guarde los datos dados por el usuario en un fichero.
- Utilizamos satisfactoriamente la función Sleep en el programa.

H. Referencias

https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=936:escribir-guardar-datos-en-ficheros-o-archivos-en-lenguaje-c-fputc-putc-fputs-fprintf-ejemplos-cu00537f&catid=82&Itemid=210

https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=935:leer-y-escribir-archivos-o-ficheros-en-c-fopen-fclose-modos-de-acceso-read-write-y-append-cu00536f&catid=82&Itemid=210