INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Práctica No. 9. Manejo de Archivos.

Unidad Temática: V. Manejo de Archivos y Puertos

Lugar de realización: Laboratorio de Cómputo **Duración:** 5.0 hrs.

Objetivo

Desarrollar un programa para la creación, manejo y cierre de archivos.

Resultados Esperados

- Trabajo de investigación logrado en el pre-reporte
- Solución de casos prácticos en lenguaje C haciendo uso de archivos, tanto archivos de texto como binarios.
- Puesta en práctica de la escritura y lectura de archivos tanto de texto como binarios y su aplicación.

Pre-reporte.

Para el buen desarrollo de la práctica el alumno entregará un trabajo previo que incluya:

1. Mostrar un ejemplo de la escritura y lectura de un archivo de texto en C.
2. Mostrar un ejemplo de la escritura y lectura de un archivo binario en C.
3. Indicar por lo menos 3 aplicaciones del uso de archivos en un programa en C.

"Hay momentos que deberían tener la opción GUARDAR".

Material y Equipo.

- IDE Dev-Cpp o compatible.
- Computadora

Introducción.

Archivos

Un **archivo** es un conjunto de datos que son almacenados en algún medio, tienen un nombre y una extensión.

Hay dos tipos de archivos:

- **Texto:** Secuencia de caracteres (.txt)
- **Binario:** Secuencia de bits (.dat, etc.)

Acceso a Archivos

Para acceder a un archivo se utiliza un **apuntador a estructura** del tipo `FILE` (`struct FILE*`), esta estructura apunta a un archivo y contiene información del archivo como:

- Tamaño del archivo
- Modo de apertura del archivo
- Nombre del archivo
- Posición actual de lectura/escritura
- Buffer

La estructura `FILE` se encuentra declarada en el archivo de cabecera:

```
stdio.h
```

Escritura de un archivo de texto

Las funciones que se utilizan para escribir texto en un archivo son:

- fopen: abre un archivo
- getchar: lee un carácter del teclado
- putc: escribe un carácter en el archivo
- fclose: cierra un archivo

Lectura de un archivo de texto

Las funciones que se utilizan para leer texto de un archivo son:

- fopen: abre un archivo
- getc: lee un carácter del archivo
- putc: escribe un carácter en el archivo
- fclose: cierra un archivo

Lectura de archivos binarios

Las funciones fprintf() y fscanf() leen y escriben variables de un archivo.

- **fprintf: escribe variables de cualquier tipo de dato estándar en un archivo.**
- **fscanf: lee variables de cualquier tipo de dato estándar de un archivo.**

Tanto para la lectura y escritura de archivos de texto y binarios, es necesario abrir el archivo usando la función `fopen`, esta función aparte de recibir como parámetro el nombre del archivo que va a abrir o crear, también recibe el modo en que se va a abrir o crear el archivo. Este modo puede ser:

- r: sólo lectura. El fichero debe existir.
- w: se abre para escritura, se crea un fichero nuevo o se sobreescribe si ya existe.
- a: apertura, se abre para escritura, el cursor se sitúa al final del fichero. Si el fichero no existe, se crea.
- r+: lectura y escritura. El fichero debe existir.
- w+: lectura y escritura, se crea un fichero nuevo o se sobreescribe si ya existe.
- a+: apertura, lectura y escritura, el cursor se sitúa al final del fichero. Si el fichero no existe, se crea.
- t: tipo texto, si no se especifica "t" ni "b", se asume por defecto que es "t"
- b: tipo binario.

Desarrollo.

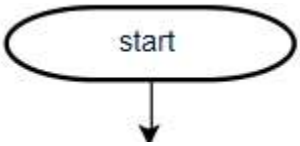
Ejemplo:

Escribir un programa que facilite el inventario de productos, el usuario deberá ingresar el código del producto, la cantidad actual del producto y la cantidad real, dichos datos serán registrados en un archivo.

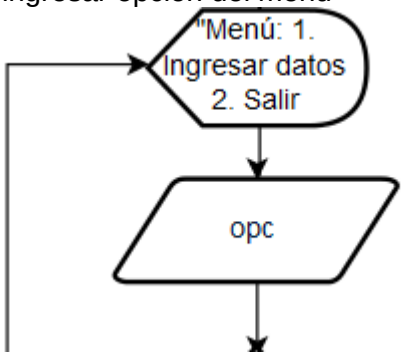
Solución:

Diagrama de Flujo

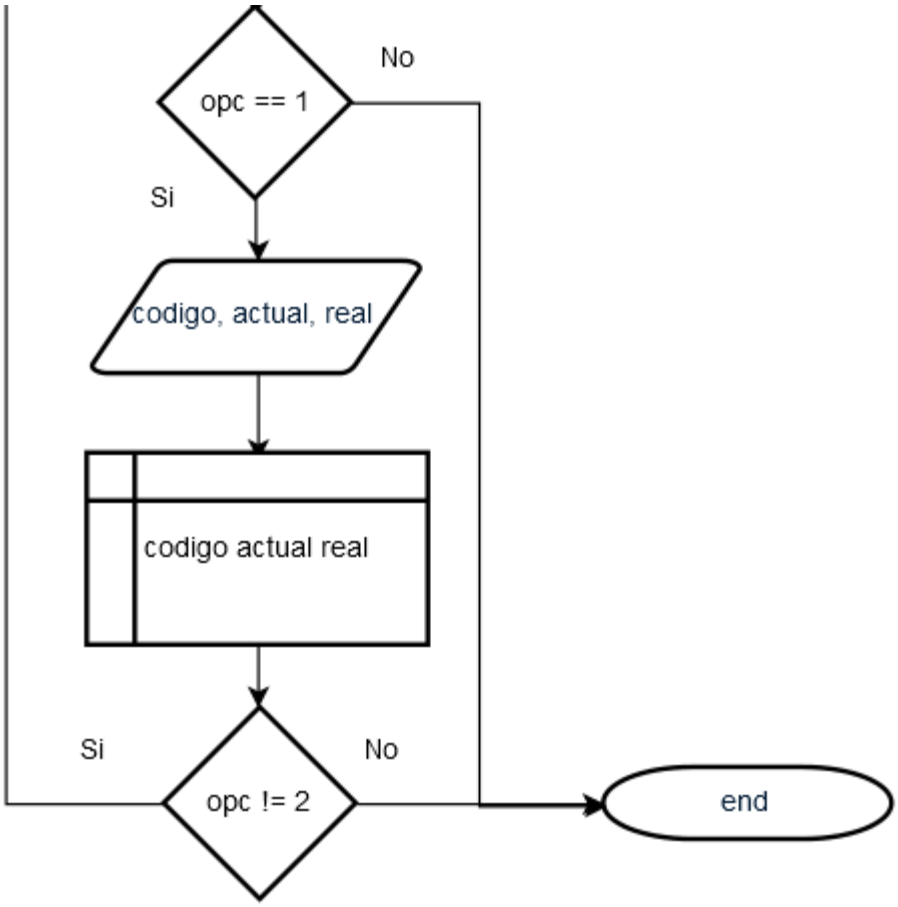
1. Inicio del programa principal



2. Mostrar el menú e ingresar opción del menú



3. Realizar condición



4. Fin

Programa en C *Ver código 9productos.c anexo en disco*

```
#include <stdio.h>
#include <conio.h>

int main()
{
char codigo[15];
int actual, real, opc;

FILE *pf;
pf = fopen("PIEZAS.DAT","bw"); //Archivo binario (contiene
                               //datos de cualquier tipo de dato)
if(pf==NULL){
     printf("ERROR EN LA OPERACION DE APERTURA");
     return 1;
}
do{
     printf("Menu:\n 1. Ingresar\n 2. Salir\n");
     scanf("%d", &opc);

     if(opc == 1){
     printf("Introduzca el código del producto, cantidad
```

```
            actual y cantidad real");

            scanf("%s %d %d", codigo, &actual, &real);
            fprintf(pf, "%s %d %d", codigo, actual, real);
            }
      }while(opc!=2);
      fclose(pf);

      getch();
      return 0;
}
```

Programas

Realice los siguientes ejercicios:

1. Los encoders producen una secuencia que puede ser usada para controlar el radio de giro, la dirección del movimiento e incluso la velocidad. Suponga un brazo robótico con hombro, codo, muñeca y mano, del que se quieren guardar los ángulos obtenidos en los ecoders cada cinco segundos para monitorear y en un futuro poder entrenar el brazo.
   Codificar un programa que permita guardar en un archivo los ángulos del hombro, codo y muñeca, de tal forma que los datos a guardar sean:

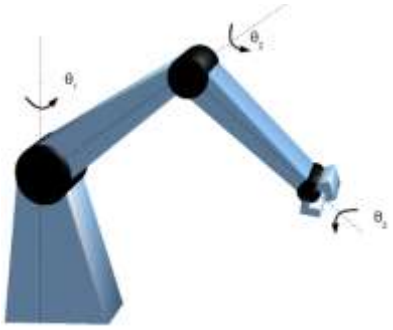


Figura 9.1 Brazo robótico

Fecha
Hora
Ángulo de Hombro
Ángulo de Codo
Ángulo de Muñeca

Notas:

- Los datos estarán contenidos en una **estructura**
- Los ángulos los generará de manera **aleatoria** cada cinco segundos en rangos específicos: el de hombro será de 0° a 140°, el del codo de 0° a 180° y el de la muñeca de 0° a 360°.
- La fecha y hora serán tomadas del **sistema**.
- Cada que se vuelva a ejecutar el programa deberá de agregar las posiciones del brazo al **final** del archivo.

Entrada:

Valores en la estructura: valores aleatorios y fecha-hora del sistema.

Procedimiento:

1) **Definir la estructura `lecturas` la cual tendrá los miembros:**
   a. **`angulo_hombro`, `angulo_codo` y `angulo_muneca` como flotantes.**
   b. **`hora` y `fecha` como cadena de caracteres**
2) **Abrir el archivo de tipo binario en modo apertura.**
3) **Crear un ciclo que repita la generación aleatoria de ángulos hasta que el usuario ya no quiera continuar (puede usar la función `kbhit()`).**
4) **Introducir los valores aleatorios y los tomados del sistema en los miembros de la estructura `lecturas`.**

5) **Escribir en el archivo con la función `fprintf()` los valores de la estructura `lecturas`, recuerde que es necesario guardar uno por uno los miembros de la estructura.**
6) **Cerrar el archivo cuando el usuario termine.**

Salida:

Archivo con los datos de los ángulos del brazo.

2. Agregar al programa anterior las siguientes funciones:
   a. Una función que permita leer del archivo todos los ángulos por lo que pasó el codo

Entrada:

Procedimiento: (existen diversas formas, aquí se muestra una pero es libre de hacerlo como guste mientras la salida sea la solicitada)

1) **Declarar una variable `angulos_leidos_de_codo`**
2) **Declarar una variable llamada `tamano_sin_codo` que contendrá la suma de los tamaños obtenidos con `sizeof` de `angulo_muneca`, `fecha`, `hora` y `angulo_hombro`**
3) **Abrir el archivo de tipo binario en modo lectura.**
4) **Colocar el cursor con `fseek` en el byte (posición) previo al ángulo del codo. (si lo hizo en el orden especificado antes están `fecha`, `hora` y `angulo_hombro`, obtenga el tamaño de estas 3 variables con `sizeof` y la suma de éstas será la posición)**
5) **Crear un ciclo que permita ir leyendo `angulo_codo`, guarde el valor en `angulos_leidos_de_codo` e imprima el contenido (para ir desplazándose incremente `fseek` con el valor dado por `tamano_sin_codo`)**
6) **Cerrar el archivo cuando termine el ciclo.**

Salida:

Ángulos de codos (en caso de que el archivo no esté vacío).

   b. Una función que permita obtener todos los ángulos en un rango de horas.

Entrada:

Valor de la variable: `hora_inicial`, `hora_final` y `dia`.

Procedimiento:

1) **Declarar la variable `hora_inicial`, `hora_final` y `dia`, y solicitar al usuario los valores**
2) **Declarar una variable (`angulos_leidos`) del tipo estructura `lecturas`**
3) **Abrir el archivo de tipo binario en modo lectura.**
4) **Crear un ciclo que permita ir leyendo estructura por estructura en el archivo y asignarlo a `angulos_leidos`**
5) **Verificar si el `dia` es igual al de `fecha` en la estructura `angulos_leidos`. En caso de que sea el día:**
   a. **Comparar si `hora_inicial` es menor al de `hora` en la estructura `angulos_leidos` y**
   b. **si `hora_final` es mayor al de `hora` en la estructura `angulos_leidos`. En caso de que estpe en el rango imprimir los datos de `angulos_leidos`**
6) **Cerrar el archivo cuando termine el ciclo.**

Salida:

Ángulos y comprendidos en la fecha y horarios establecido (en caso de ser encontrado en el archivo).

c. Una función para ver las condiciones del brazo cuando el hombro se estiró cerca del máximo (por ejemplo: ángulo mayor a 120°)

Entrada:

Valor de la variable: `angulo_maximo`.

Procedimiento:

7) **Declarar la variable `angulo_maximo` y asignarle un valor**
8) **Declarar una variable (`angulos_leidos`) del tipo estructura `lecturas`**
9) **Abrir el archivo de tipo binario en modo lectura.**
10) **Crear un ciclo que permita ir leyendo estructura por estructura en el archivo y asignarlo a `angulos_leidos`**
11) **Verificar si `angulo_hombro` de `angulos_leidos` es mayor que `angulo_maximo`. En caso de que sea mayor, entonces imprimir `angulos_leidos`**
12) **Cerrar el archivo cuando termine el ciclo.**

Salida:

Ángulos y fechas en las que el ángulo del hombro estuvo cerca del máximo (en caso de ser encontrado en el archivo).

3. Agregue al programa un menú en el que el usuario pueda seleccionar si quiere hacer lectura de los ángulos o si desea alguna consulta del punto anterior.

Entrada:

Valor de la variable: `opcion`.

Procedimiento:

1) **Declarar la variable `opcion`.**
2) **Escribir un ciclo que se repita hasta que el usuario seleccione salir**
3) **En el ciclo se imprimirá el menú y el usuario seleccionará una opción que permitirá el llamado a las funciones**

Salida:

Función solicitada

Problema de Aplicación

Haciendo uso de estructura y archivos escriba un programa que

Monitorización cardiaca. El uso de monitores nos permite controlar las funciones vitales y complementan nuestra función, para ello se evalúan 3 parámetros: frecuencia cardiaca, frecuencia respiratoria y pulso. Se requiere guardar los datos recopilados. Posteriormente se abrirá el archivo y analizará de tal forma que despliegue la hora y el parámetro que salió del rango normal, se determinará si el paciente muestra más anomalías por la mañana (6am-11am), por la tarde (11am-6pm) o por la noche(6pm-6am), y se dirá cuál de los factores tuvo más anomalías.

Propuesta de Solución. Se sugieren los siguientes aspectos:

- Use archivos binarios
- Suponga tres estructuras (una para cada parámetro) para que sean guardados y leídos de una manera más sencilla y eficaz
- Simule el monitoreo cada 10 segundos y con entrada simuladas con un random, considera que en alguno de los valores debe estar fuera del rango válido

Proyectos (opcionales).

1. Guardar una base de datos en archivos, por ejemplo para la base de datos de un Banco cada tabla estaría representada por un archivo, se tendrían las tablas: Cliente, Cuenta, Banco, etc y cada una sería un archivo donde estarían guardados los datos correspondientes.

2. Generar la base de datos anterior pero usando estructuras (struct), las variables de las estructuras son las que se guardarían en los archivos binarios que representarían a cada tabla de la base de datos.

Ponderación de la Práctica.

Sección	Elemento a Evaluar
Pre-reporte	Ejemplos de archivos y sus aplicaciones
Ejercicio 1	Aplicación de estructuras en archivos binarios
Ejercicio 2a	Búsqueda en archivos con la función `fseek`
Ejercicio 2b	Búsqueda en archivos desplazándose estructura por estructura
Ejercicio 2c	Búsqueda en archivos desplazándose estructura por estructura
Ejercicio 3	Creación de menú y llamadas a función
Problema	Archivos binarios y estructuras

Bibliografía.

Deitel P.J.y Deitel H. M., Como Programar C++, Ed. Prentice Hall, 6 ª Impresión, México, 2009, ISBN: 970-26-1273-X, Págs: 1-1050.

Deitel P.J. y Deitel H. M., Como programar en C#, Ed. Prentice Hall, 2ª Impresión, México, 2007,ISBN: 9702610567, Págs: 1-1080.

Guardati, Silvia, Estructura de Datos Orientado a Objetos con C ++, Ed.Prentice, México 2007, ISBN: 9702607922, Págs: 1–183.

Cairó Battistuti, Fundamentos de Programación. Piensa en C, Ed. Prentice Hall, México 2006, ISBN: 970-26-0810-4

Antonakos James, Mansfiel Kenneth, Programación Estructurada en C, Ed. Prentice Hall, Madrid 1997, ISBN: 0-13-520487-9

NOTA: Presentar el reporte en un documento (PDF o DOC), el código fuente y la impresión de la pantalla de ejecución.

"Puedes apoyarte de tus compañeros y profesor para aclarar tus dudas"