|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | RODRIGUES ESPINO CLAUDIA |
| *Asignatura:* | FUNDAMENTOS DE PROGRAMACION |
| *Grupo:* | 4 |
| *No de Práctica(s):* | PRACTICA 13: Lectura y escritura de datos. |
| *Integrante(s):* | FLORES MENDOZA OLGA |
|  |  |
| *No. de Equipo de cómputo empleado* | 44 |
| *Semestre:* | 2019-2 |
| *Fecha de entrega:* | 16 DE MAYO DE 2019 |
| *Obervaciones:* |  |
|  |  |

Archivo

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**PRÁCTICA 13**

**OBJETIVOS:**

Elaborar programas en lenguaje C que requieran el uso de archivos de texto plano en la resolución de problemas, entendiendo a los archivos como un elemento de almacenamiento secundario.

**ACTIVIDADES:**

* A través de programas en C, emplear las funciones para crear, leer, escribir y sobrescribir archivos de texto plano.
* Manipular archivos empleando los diferentes tipos de acceso a ellos.

**INTRODUCCIÓN:**

Un archivo es un conjunto de datos estructurados en una colección de entidades elementales denominadas registros que son del mismo tipo, pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Lenguaje C permite manejar la entrada y la salida de datos desde o hacia un archivo.

**Apuntador a archivo**

Un apuntador a un archivo es un hilo común que unifica el sistema de Entrada/Salida (E/S) con un buffer donde se transportan los datos. Señala la información que contiene y define el nombre, el estado y la posición actual del archivo. En lenguaje C los apuntadores se manejan como variables apuntador de tipo FILE que se define en la cabecera *stdio.h*. La sintaxis para obtener una variable apuntador de archivo es la siguiente:

**Abrir archivo**

La función fopen() abre una secuencia para que pueda ser utilizada y la asocia a un archivo. Su estructura es la siguiente:

**\*FILE fopen(char \**nombre\_archivo*, char \**modo*);**

Donde *nombre\_archivo* es un puntero a una cadena de caracteres que representan un nombre valido del archivo y puede incluir una especificación del directorio. La cadena a la que apunta *modo* determina como se abre el archivo. Existen diferentes modos de apertura de archivos, los cuales se mencionan a continuación:

* r: Abre un archivo de texto para lectura.
* w: Crea un archivo de texto para escritura.
* a: Abre un archivo de texto para añadir.
* r+: Abre un archivo de texto para lectura / escritura.
* w+: Crea un archivo de texto para lectura / escritura.
* a+: Añade o crea un archivo de texto para lectura / escritura. rb: Abre un archivo en modo lectura y binario.
* wb: Crea un archivo en modo escritura y binario.

**Cerrar archivo**

La función fclose() cierra una secuencia que fue abierta mediante una llamada a fopen(). Un error en el cierre de una secuencia puede generar todo tipo de problemas, incluyendo la perdida de datos, destrucción de archivos y posibles errores intermitentes en el programa. La firma de esta función es:

**int fclose(FILE \*apArch);**

**Funciones fgets y fputs**

Las funciones fgets() permite leer una cadena desde el archivo especificado. Esta función lee un renglón a la vez.

**char \*fgets(char \*buffer, int tamaño);**

La función fputs() permite escribir una cadena en un archivo especifico

**FILE \*apArch); char \*fputs(char \*buffer, FILE \*apArch);**

**Funciones fscanf y fprintf**

Las funciones fprintf() y fscanf() se comportan exactamente como printf() (imprimir) y scanf() (leer), excepto que operan sobre archivo. Sus estructuras son:

**int fprintf(FILE \**apArch*, char \**formato*, ...);**

**int fscanf(FILE \**apArch*, char \**formato*, ...);**

Donde *formato* es una cadena que puede incluir texto o especificadores de impresión de variables. En los puntos suspensivos se agregan las variables (si es que existen) cuyos valores se quieren escribir en el archivo.

**Funciones fread y fwrite**

fread permite leer uno o varios elementos de la misma longitud a partir de una dirección de memoria determinada (apuntador).

El valor de retorno es el número de elementos (bytes) leídos. Su sintaxis es la siguiente:

**int fread(void \*ap, size\_t tam, size\_t nelem, FILE \*archivo)**

fwrite permite escribir hacia un archivo uno o varios elementos de la misma longitud almacenados a partir de una dirección de memoria determinada.

El valor de retorno es el número de elementos escritos. Su sintaxis es la siguiente:

**int fwrite(void \*ap, size\_t tam, size\_t nelem, FILE \*archivo)**

**DESARROLLO DE ACTIVIDADES:** Los siguientes programas fueron escritos en la terminal, GNU, todos son la finalidad de hacer uso (Leer, escribir, sobrescribir) de archivos.

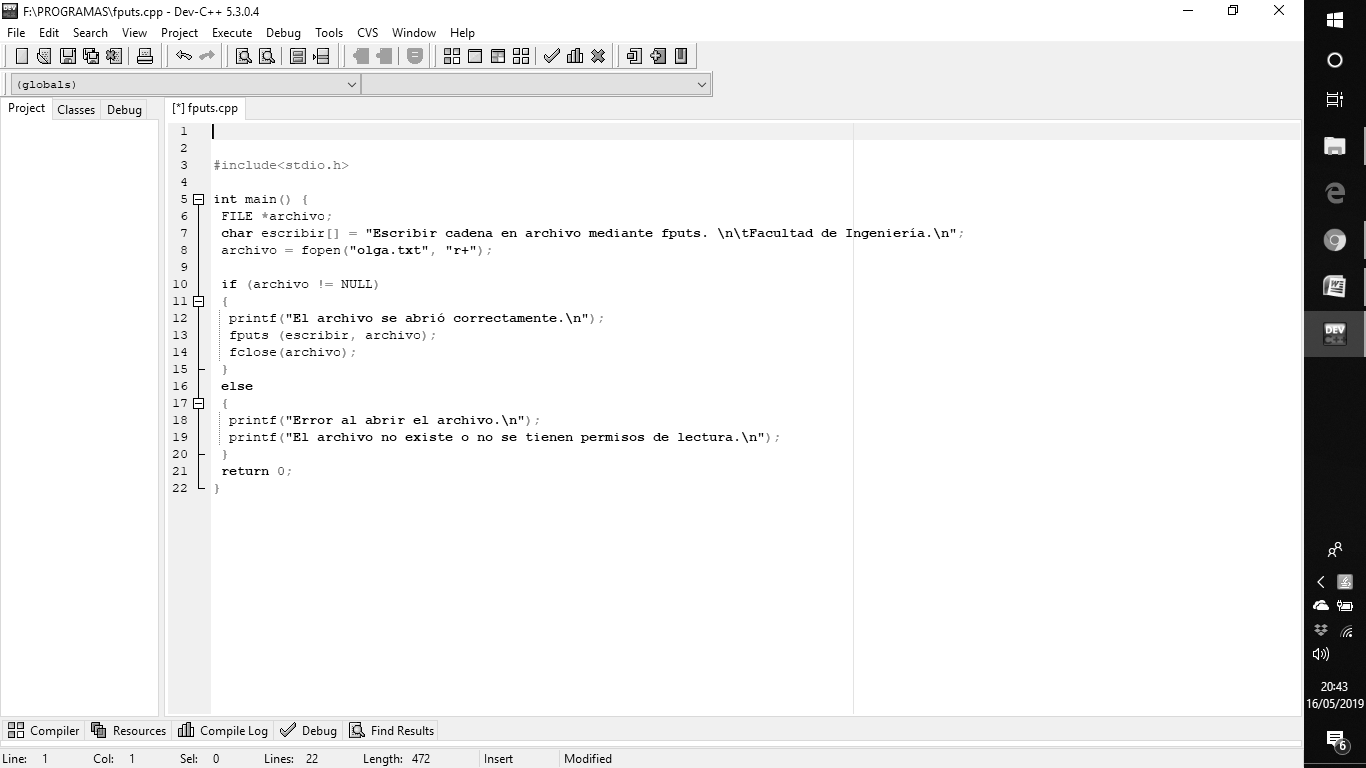
En todos los programas existió un erro al momento de compilar, el cual fue a causa de que el archivo no había sido creado.

Al principio se pensó que el programa iba a crear el archivo directamente, pero al existir esto errores se entendió que se debía de crear por fuera cada archivo, por esta razón se creó primero el archivo y posteriormente se escribió el nombre de este en le editor de lenguaje, eso se muestra a continuación.

**PROGRAMA 1:** Este programa escribir una proposición en un archivo, empleando fputs, y las estructura de selección if-else.

El archivo usado se llama olga.txt es importante agregarle esta terminación (.txt) ya que esta usada en la terminal indica que se crea y se trata de un archivo con modo r+: Abre un archivo de texto para lectura / escritura. El apuntador se llama **archivo**.

Cuando la tarea no se puede realizar aparecerá lo que se indica el if-else

****

Apuntador.

Archivo

Proposición

Nota: En día de la práctica se olvido tomar la foto del programa en la terminal usando nano es por esto que se uso la foto del programa escrito en Dev-c++. Pero existieron los mismos acontecimientos.

A continuación se muestra con lo que sucedió con este programa:

1. Cuando se realizado la primera ejecución apareció **Error al abrir el archivo** esto sucedió porque el archivo aun no había sido creado. Pero todavía no se encontraba la razón.
2. En este caso ocurrió lo mismo, pero aquí ya se entendió que sucedía y como debía remediarse.

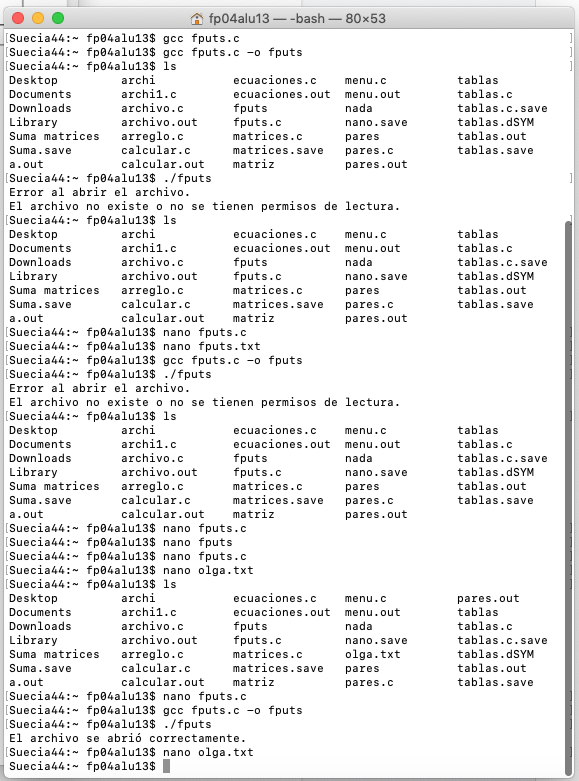
La solución era crear al archivo por fuera (Del editor) y así mencionarlo en el programa para que funcionara correctamente.

Para crear el archivo se uso la sintaxis

**nano nombre.txt**

De esta manera se surgió el archivo.

Posteriormente se compilo el programa y haciendo esto ocurrió el numero 3.

****

Creación del archivo.

2

1

3

1. Aquí aparece **El archivo se abrió correctamente**. Y esto significa que el programa ya no tiene errores y realizara correctamente la ejecución.

Se entro al nano del archivo y aparase escrita la continuación. Es decir, se comprueba la ejecución del programa.

****

**PROGRAMA 2:** Este programa escribe una matriz M 3x3 en un archivo con los valores que da el usuario.

Se empleo un puntador el cual es nombrado **acb**, un archivo llamado **matriz.txt**, también se uso un arreglo para realizar la matrizy la estructura de repetición for. El modo de este archivo fue w: Crea un archivo de texto para escritura.

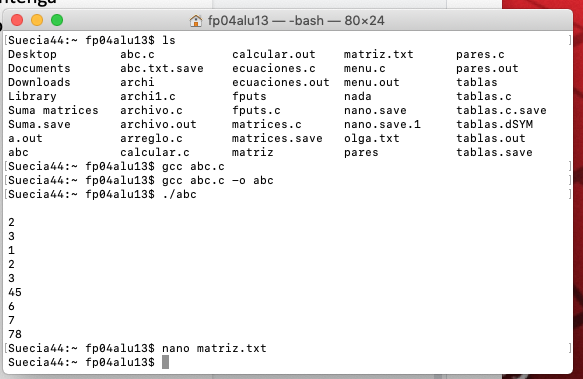
****

Arreglo

Archivo.

Apuntador.

Antes de compilar el programa se creó el archivo, posteriormente se realizo la compilación y al no haber errores pero prosiguió a realizar la ejecución. Esto se muestra a continuación.

****

Valores.

Archivo.

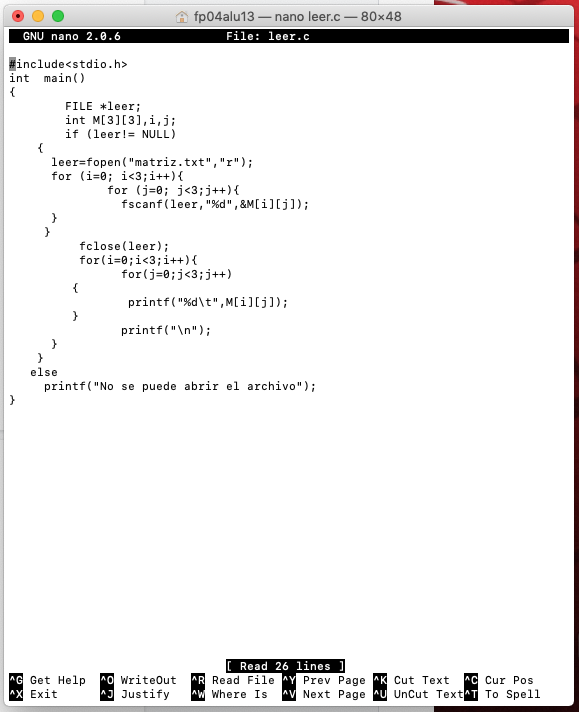
El archivo muestra la matriz con los valores que dio el usuario.

****

**PROGRAMA 3:** Este programa lee desde un archivo la matriz M de 3x3 de tipo entero. Y muestra en pantalla la matriz M.

En este caso se utilizo un apuntador llamado **leer**, un arreglo para la matriz,la estructura for y el archivo **matrix.txt** el cual es el mismo del programa anterior. Pero en este programa cambio el modo a r: Abre un archivo de texto para lectura.

Como en el programa anterior se escribió la matriz en el archivo lo único que hará este programa será leer la matriz que se encuentra en el archivo mencionado.

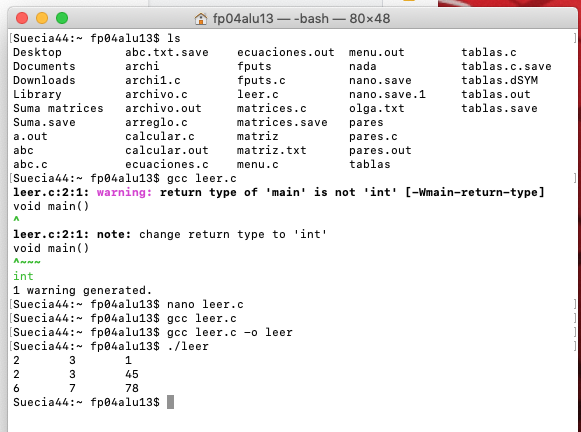
****

Archivo

Arreglo

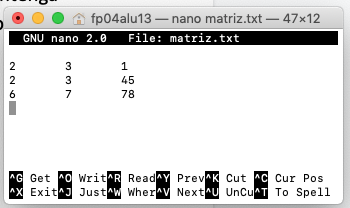
Apuntador

Se verifica que el programa se haya guardo y prosigue la compilación, al realizarla aparece un error el cual es corregido sin ningún problema. Se realiza una nueva compilación que no muestra errores y posteriormente la ejecución. Como se puede observar el programa arroja los valores contenidos en el archivo matriz.txt realizando su tarea correctamente.

****

Matriz contenida en el archivo.

Si se entra al archivo se encuentra la matriz M 3x3 intacta.

****

**CONCLUSION:**

En esta práctica se llevo a cabo el uso de los archivos, es un tema difícil, ya que en proceso de editar los programas existen confusiones, como son ¿Cómo hacer el archivo? ¿Cómo hacer el apuntado? El programa no crea automáticamente el archivo sino que este debe de ser creado de manera externa al programa, una vez creado el archivo se debe hacer mención el programa con el que se quiere relacionar. De lo contario el editor no reconoce al archivo. Es importante no olvidar el modo en el que se abrirá el archivo, ya que esto determina la acción del archivo y su acceso. Se cumple el objetivo y las actividades ya que el uso de los archivos fue para leer y escribir datos en ellos.