

Contenido

[**Objetivo:** 3](#_Toc530427467)

[**Actividades:** 3](#_Toc530427468)

[**Introducción** 3](#_Toc530427469)

[**Apuntador a archivo** 3](#_Toc530427470)

[**Abrir archivo** 3](#_Toc530427471)

[**Cerrar archivo** 4](#_Toc530427472)

[**Funciones fgets y fputs** 5](#_Toc530427473)

[**Funciones fscanf y fprintf** 5](#_Toc530427474)

[**Funciones fread y fwrite** 5](#_Toc530427475)

[**Desarrollo** 6](#_Toc530427476)

[**Conclusiones:** 12](#_Toc530427477)

[**Bibliografías:** 12](#_Toc530427478)

# **Objetivo:**

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 travéz 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**

## **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.

Un apuntador a archivo señala a la información que contiene y define ciertas

características sobre él, incluyendo el nombre, el estado y la posición actual del archivo.

Los apuntadores a un archivo se manejan en lenguaje C 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**

abre una secuencia para que pueda ser utilizada y la asocia a un archivo. Su prototipo es:

FILE \*fopen(const char nombre\_archivo, cost charmodo);

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. La siguiente tabla muestra los valores permitidos para modo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Modo / Significado | | r Abre un archivo de texto para lectura. | | w Crea un archivo de texto para escritura. | | a Abre un archivo de texto para añadir. | | rb Abre un archivo binario para lectura. | | wb Crea un archivo binario para escritura. | | ab Abre un archivo binario 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. | | r+b Abre un archivo binario para lectura / escritura. | | w+b Crea un archivo binario para lectura / escritura. | | a+b Añade o crea un archivo binario para lectura / escritura. | |

//aperturaDeArchivos

#include<stdlib.h>

#include<stdio.h>

Int main()

{

FILE\*fp;

Fp=open(“archivo.txt”,”r”);

If(fp==NULL)

}

printf(“Error al abrir el archivo para leer”);

Exit(1);

return 0;

## **Cerrar archivo**

**Fclose**

Devuelve un puntero a archivo. Un programa nunca debe alterar el valor de ese puntero. Si se produce un error cuando se está intentando abrir un archivo, fopen() devuelve un puntero nulo.

Se puede abrir un archivo bien en modo texto o binario. En la mayoría de las implementaciones, en modo texto, la secuencias de retorno de carro / salto de línea se convierten a caracteres de salto de línea en lectura. En la escritura, ocurre lo contrario: los caracteres de salto de línea se convierten en salto de línea. Estas conversiones no ocurren en archivos binarios.

La macro NULL está definida en STDIO.H. Este método detecto cualquier error al abrir un archivo: como por ejemplo disco lleno o protegido contra escritura antes de comenzar a escribir en él.

Cierre de un archivo.

La función fclose() cierra una secuencia que fue abierta mediante una llamada a fopen(). Escribe toda la información que todavía se encuentre en el buffer en el disco

y realiza un cierre formal del archivo a nivel del sistema operativo. Un error en el cierre de una secuencia puede generar todo tipo de problemas, incluyendo la pérdida de datos, destrucción de archivos y posibles errores intermitentes en el programa. El prototipo de esta función es:

int fclose(FILE \*F);

Donde F es el puntero al archivo devuelto por la llamada a fopen(). Si se devuelve un valor cero significa que la operación de cierre ha tenido éxito. Generalmente, esta función solo falla cuando un disco se ha retirado antes de tiempo o cuando no

queda espacio libre en el mismo.

## **Funciones fgets y fputs**

La función fgets() lee una cadena desde el archivo especificado hasta que lee un carácter de nueva línea o longitud-1 caracteres.

char \*fgets(char \*str, int long, FILE \*F);

Si se produce un EOF (End of File) la función gets retorna un NLa función puts() escribe la cadena a un archivo especifico.

char \*fputs(char \*str, FILE \*F); ULL.

## **Funciones fscanf y fprintf**

Estas funciones se comportan exactamente como prinft() y scanf() discutidas anteriormente, excepto que operan sobre archivo. Sus prototipos son:

int fprintf(FILE \*F, const char \*cadena\_de\_control, .....);

int fscanf(FILE \*F, const char \*cadena\_de\_control, .....);

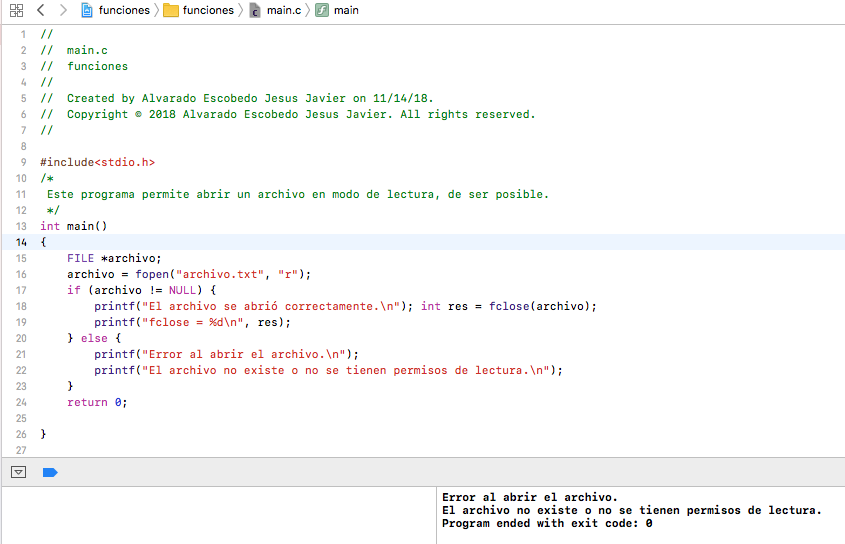
Donde F es un puntero al archivo devuelto por una llamada a fopen(). fprintf() y

fscanf() dirigen sus operaciones de E/S al archivo al que apunta F.

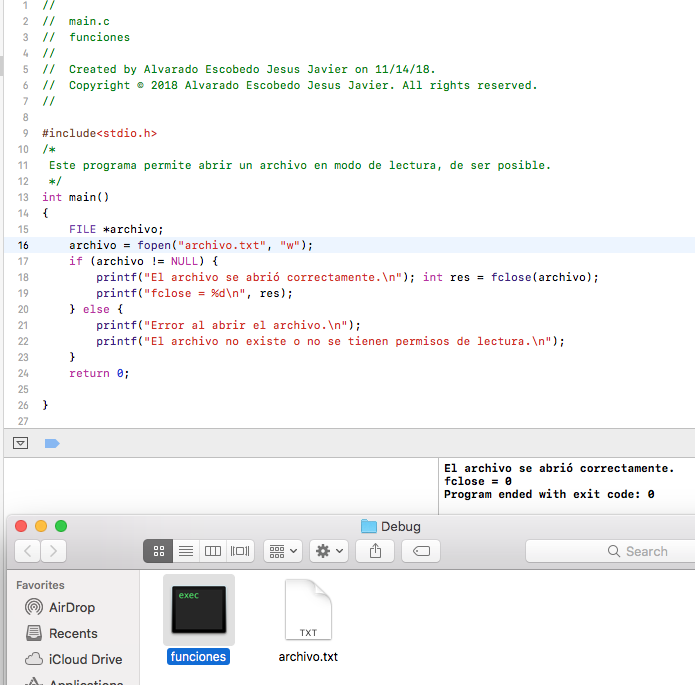
## **Funciones fread y fwrite**

Esta función lee un bloque de una "stream" de datos. Efectúa la lectura de un arreglo de elementos "count", cada uno de los cuales tiene un tamaño definido por "size". Luego los guarda en el bloque de memoria especificado por "ptr". El indicador de posición de la cadena de caracteres avanza hasta leer la totalidad de bytes. Si esto es exitoso la cantidad de bytes leídos es (size\*count).

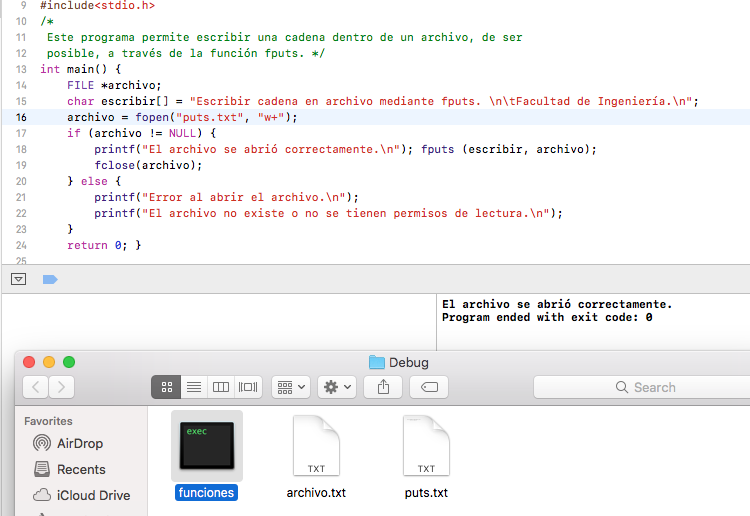
# **Desarrollo**

****

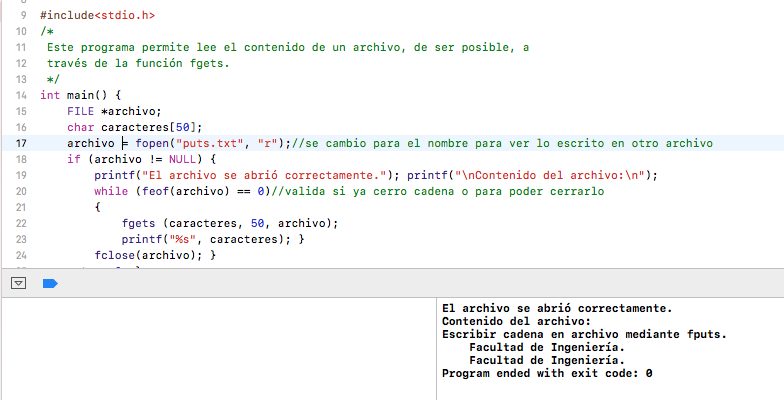
Como el archivo no habría porque no existía teníamos que cambiar el modo de fopen cambiar de r a w y sale lo siguiente:



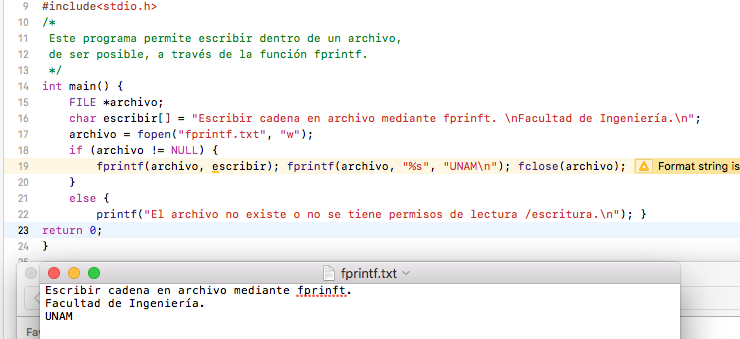
**fgets y fputs**

****

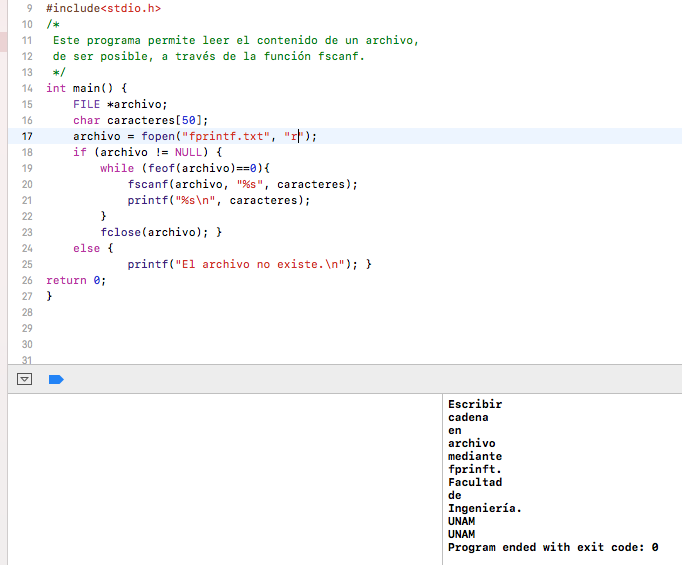
**fputs**

****

**fprintf**

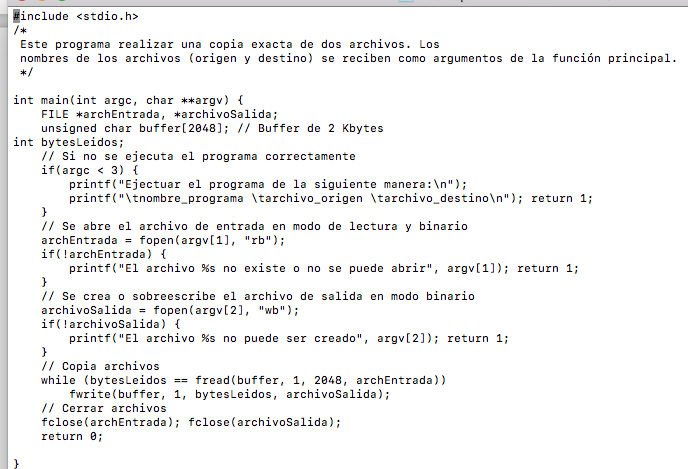
****

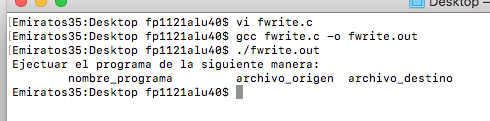
**fscanf**

****

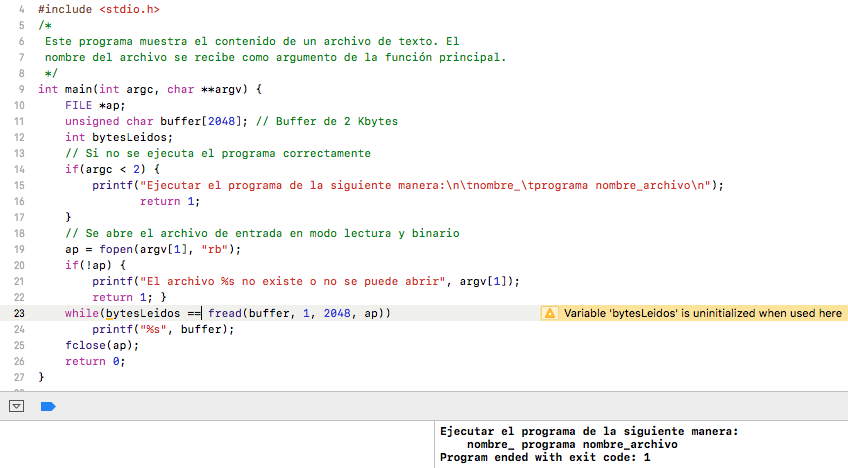
Comparar fprintf y fscanf:

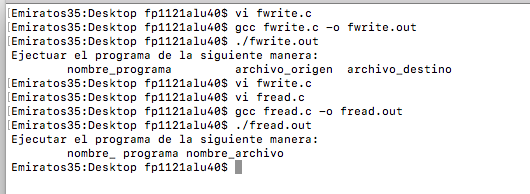
fwrite



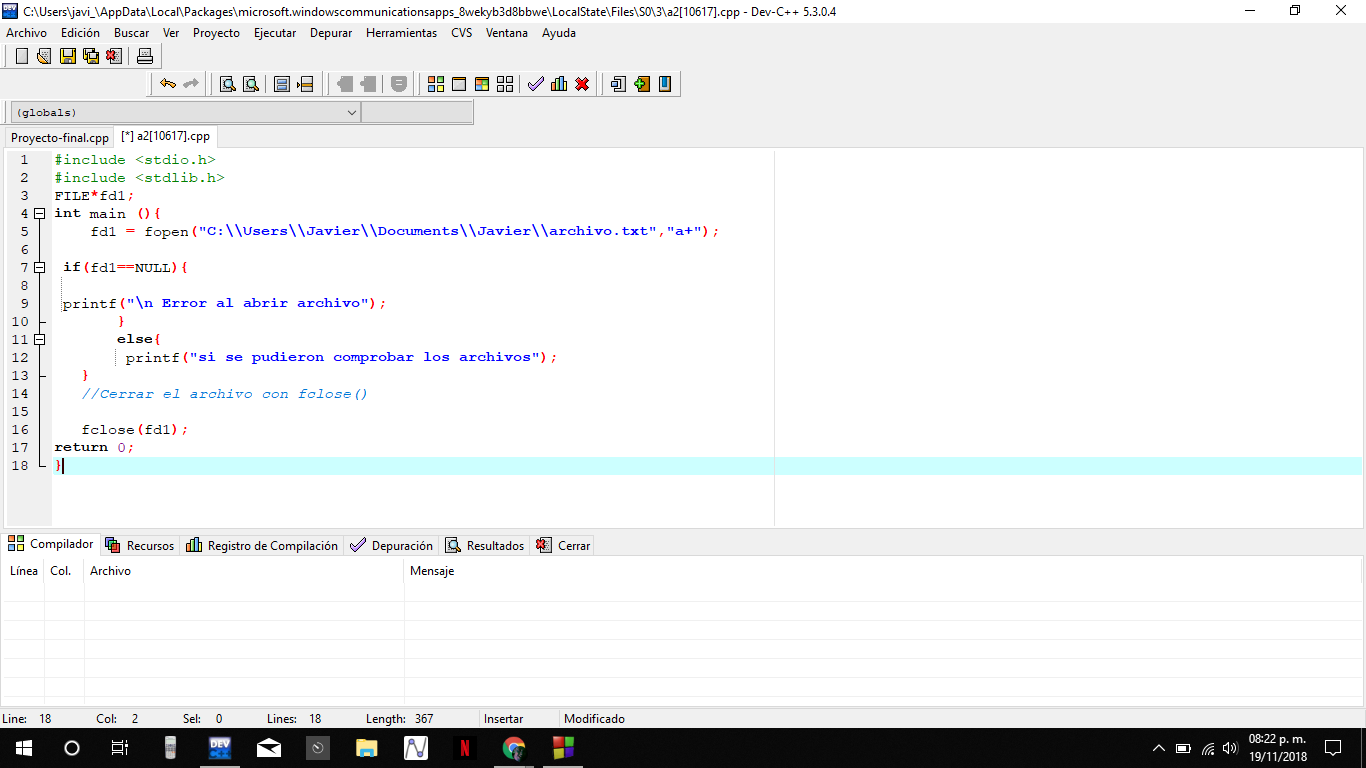


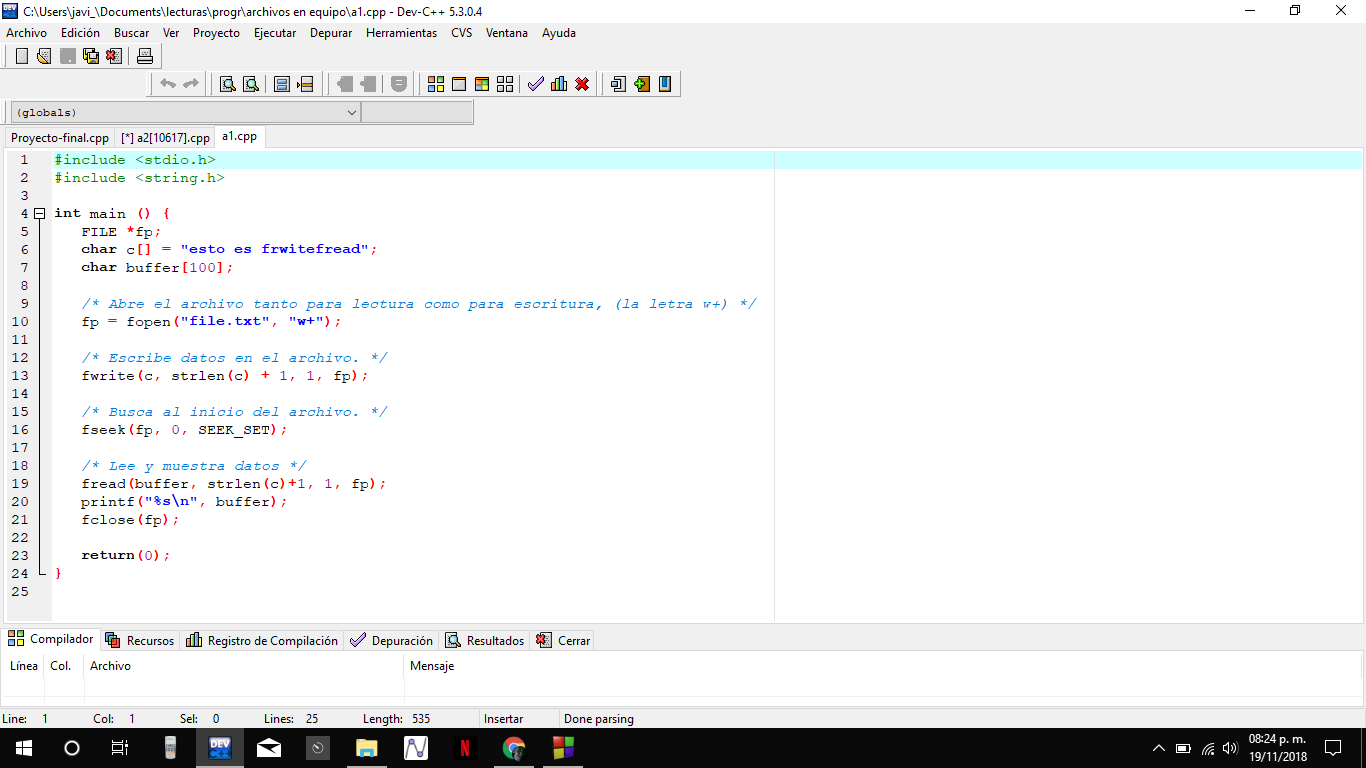
**fread**

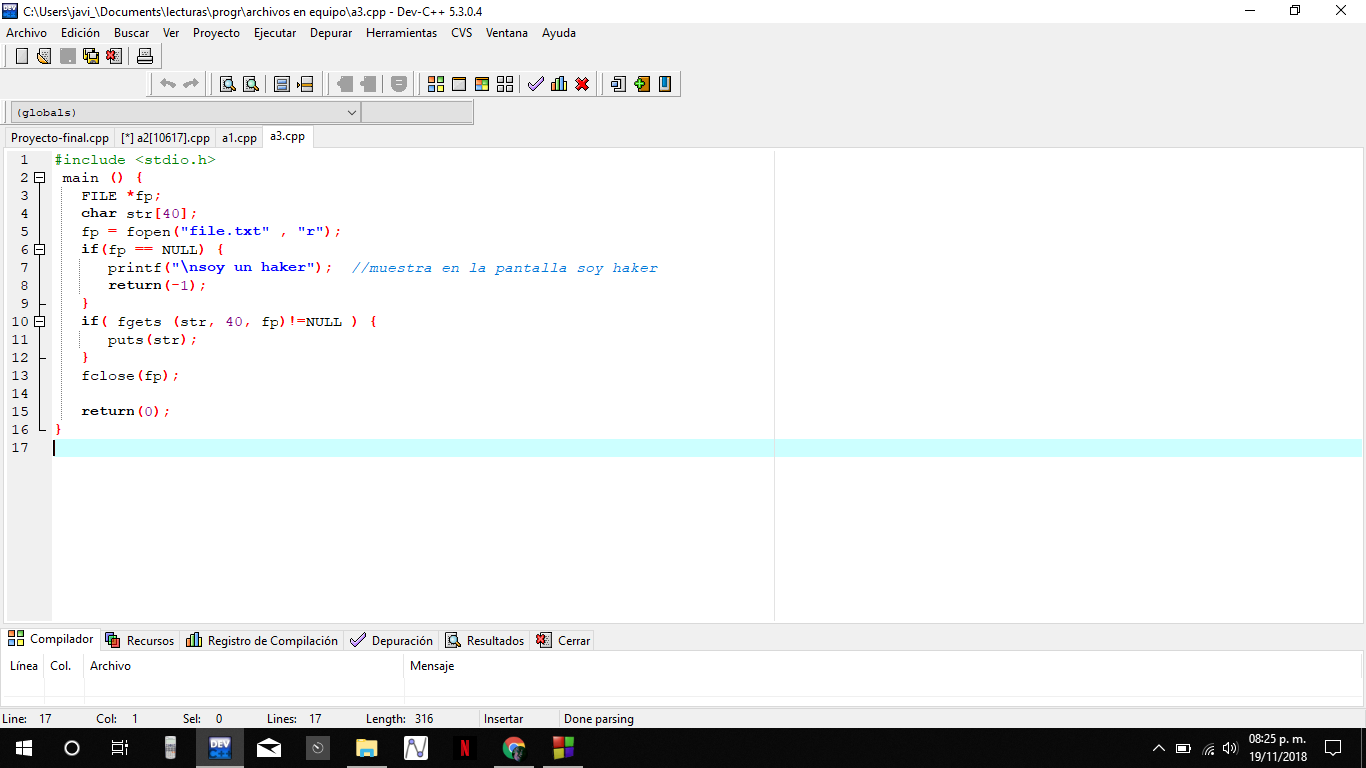
****

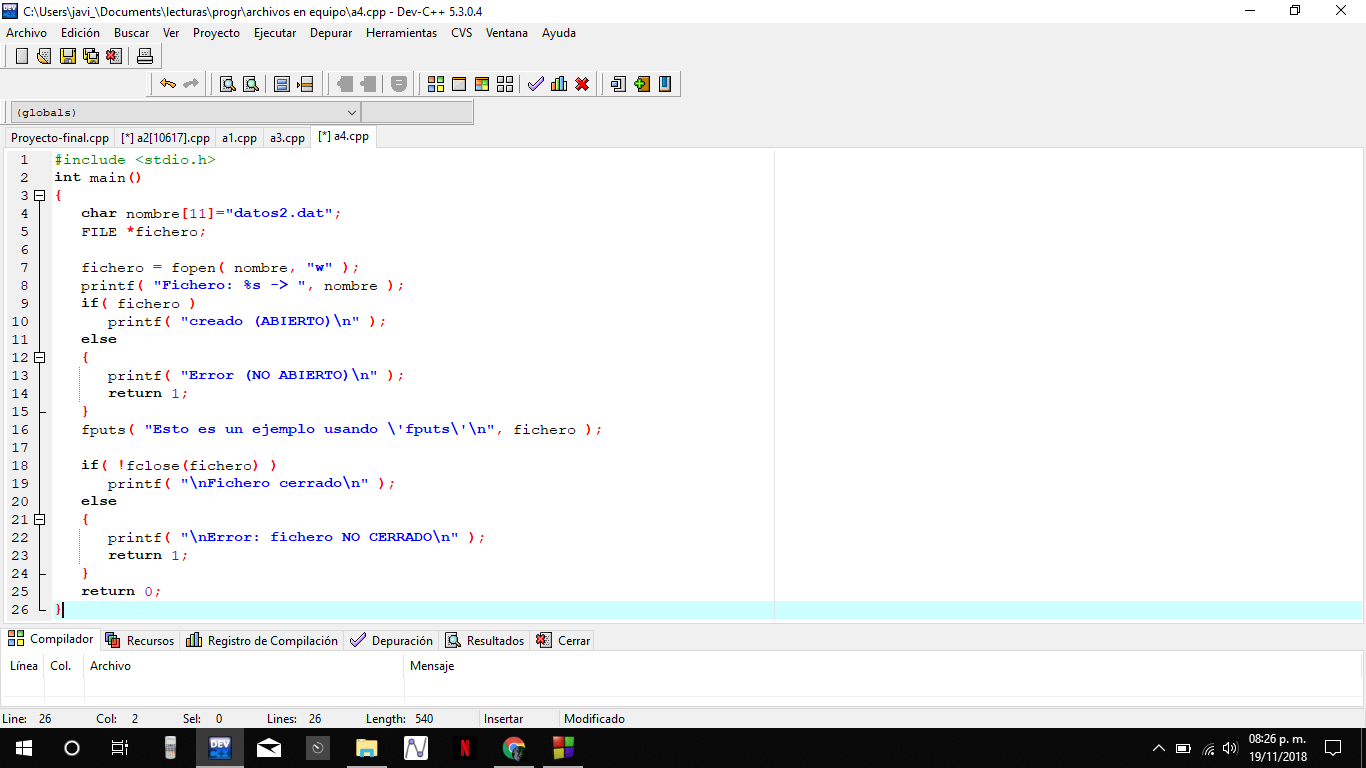
****

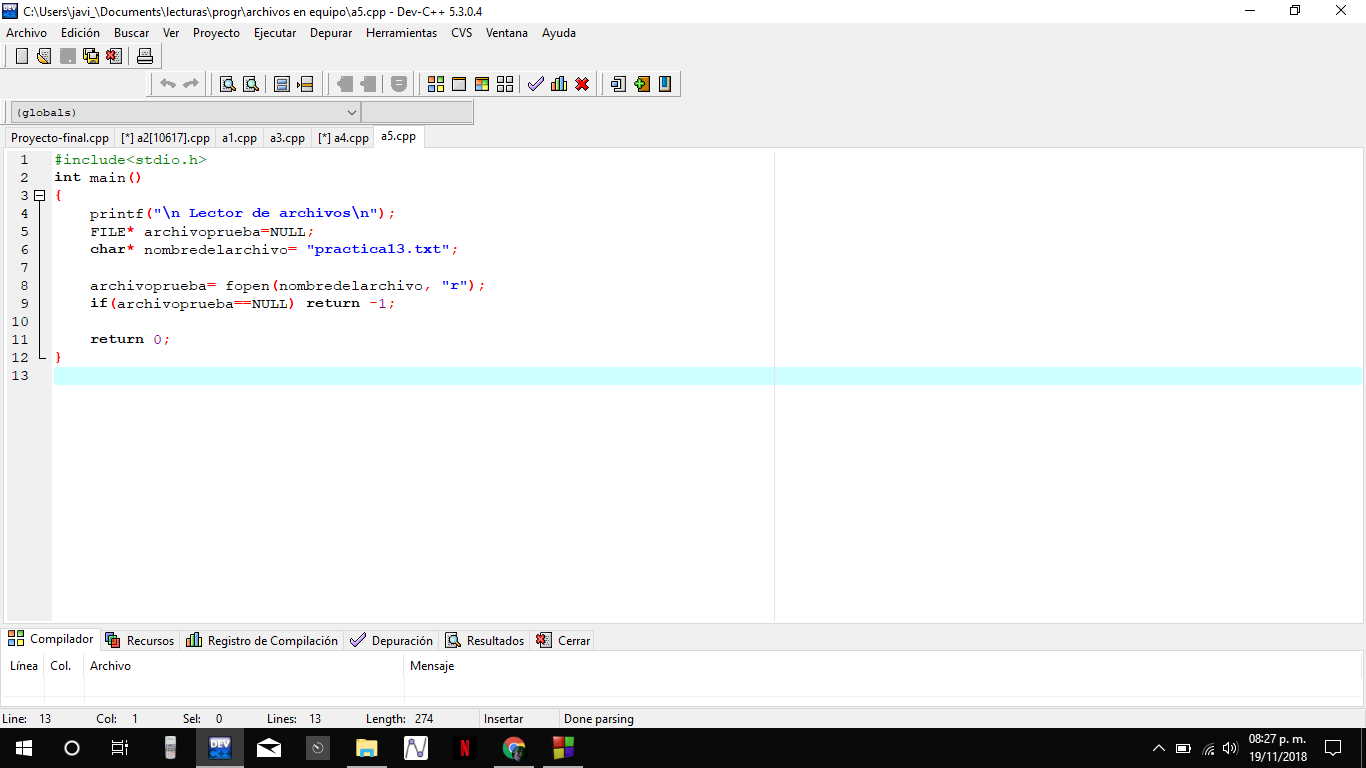
**Trabajo en equipo.**











# **Conclusiones:**

**En fclose y fopen prácticamente su uso es paralelo, y fputs y gets se tiene que tener cuidado su uso si de acuerdo a nuestras necesidades si creamos el archivo o lo modificamos**

# **Bibliografías:**

El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.