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

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. Claudia Rodríguez Espino |
| *Asignatura:* | Fundamentos de Programación |
| *Grupo:* | 1102 |
| *No de Práctica(s):* | 13 |
| *Integrante(s):* | Alejandro Meneses Mercado |
|  |  |
|  |  |
| *Semestre:* | 2018-1 |
| *Fecha de entrega:* | 17 de noviembre del 2017 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

Práctica 13: Lectura y escritura de datos

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.

Desarrollo:

Primero vimos que un archivo es un conjunto de datos estructurados en una colección de entidades denominadas registros que son del mismo tipo, pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

También vimos que el lenguaje C permite manejar la entrada y salida de datos de un archivo mediante la biblioteca stdio.h.

Después vimos que 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.

Vimos que en lenguaje C estos apuntadores se manejan como variables apuntador de tipo FILE: Su sintaxis es FILE \*F;

Aparte vimos como abrir un archivo mediante la función fopen() y su sintaxis es: \*FILE fopen(char \*nombre\_archivo, char \*modo);

Donde el nombre del archivo es un puntero a una cadena de caracteres que representan un nombre válido del archivo y modo determina como se abre el archivo.

Revisamos mas a profundidad lo que era el modo y vimos algunos y los mas comunes los cuales son:

r: Abre un archivo de texto para lectura.

w: Crea un archivo de texto para escritura.

r+: Abre un archivo de texto para lectura / escritura.

w+: Crea un archivo de texto para lectura / escritura.

También vimos como cerrar un archivo con la función fclose() que realiza un cierre formal del archivo a nivel del sistema operativo.

Vimos que 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.

Su sintaxis es: int fclose(FILE \*apArch);

Donde apArch es el apuntador al archivo devuelto por la llamada a fopen().

Vimos que esta función solo falla cuando un disco se ha retirado antes de tiempo o cuando no queda espacio libre en el mismo.

Después vimos las funciones fgets y fputs las cuales pueden leer y escribir, respectivamente, cadenas sobre los archivos.

También vimos que la función fputs() permite escribir una cadena en un archivo especifico y la función fgets() permite leer una cadena desde el archivo especificado.

Aparte vimos las funciones fscan y fprint se comportan exactamente como printf()y scanf(), excepto que operan sobre archivo.

Y su sintaxis es: int fprintf(FILE \*apArch, char \*formato, ...);

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

Donde apArch es un apuntador al archivo devuelto por una llamada a la función fopen() y el formato es una cadena que puede incluir texto o especificadores de impresión de variables.

Como actividad agregamos 5 programas vistos en la práctica:

1. Programa que abre y cierra un archivo

#include<stdio.h>

int main()

{

FILE \*archivo;

archivo = fopen("archivo.txt", "w");

if (archivo != NULL)

{

printf("El archivo se abrio correctamente.\n");

int res = fclose(archivo);

printf("fclose = %d\n", res);

} else

{

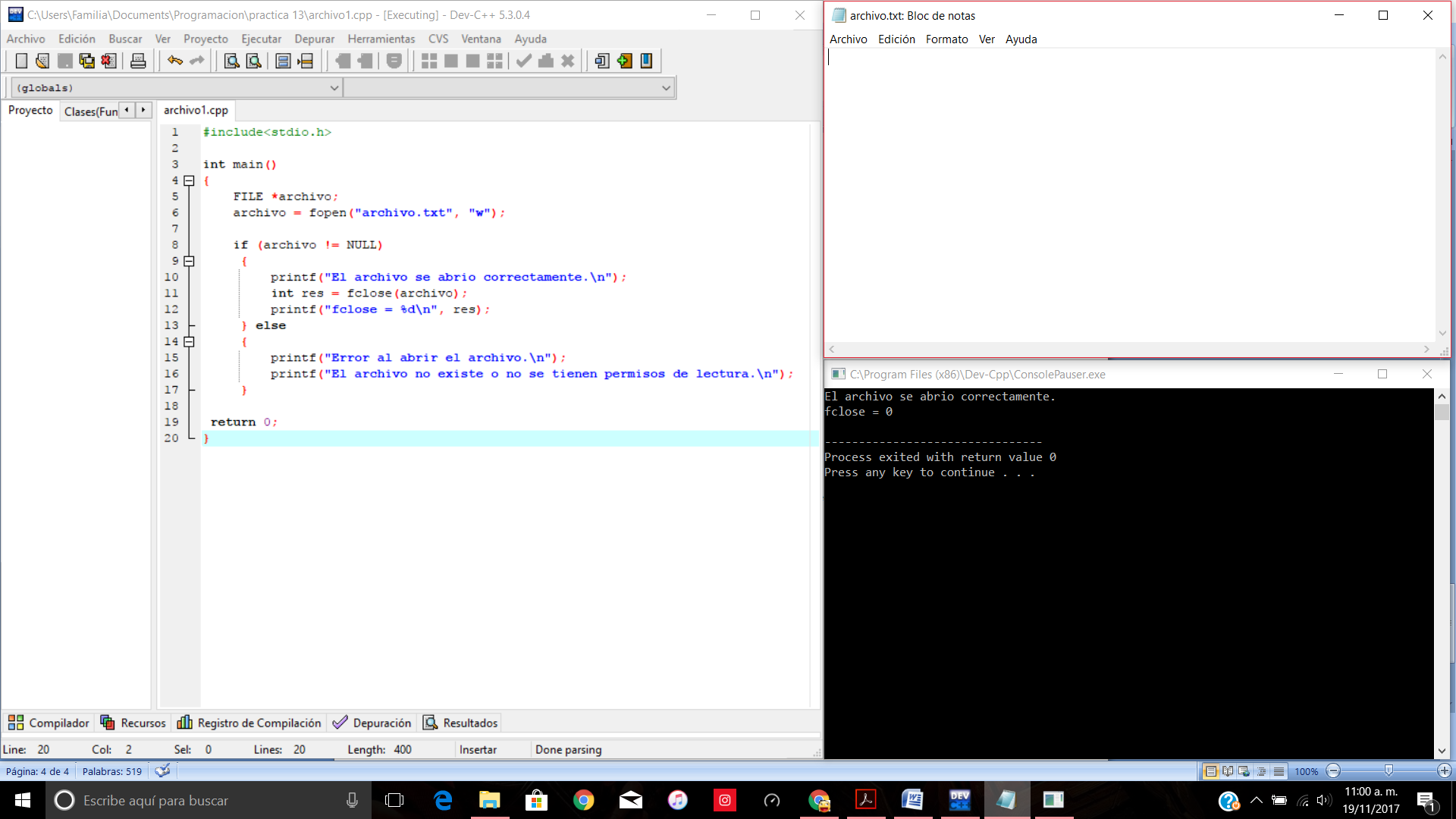
printf("Error al abrir el archivo.\n");

printf("El archivo no existe o no se tienen permisos de lectura.\n");

}

return 0;

}



1. Programa que utiliza la función fputs

#include<stdio.h>

int main()

{

FILE \*archivo;

char escribir[] = "Escribir cadena en archivo mediante fputs. \n\tFacultad de Ingeniería.\n \tFundamentos de Programación\n \tUNAM";

archivo = fopen("puts.txt", "w+");

if (archivo != NULL)

{

printf("El archivo se abrio correctamente.\n");

fputs (escribir, archivo);

fclose(archivo);

}

else

{

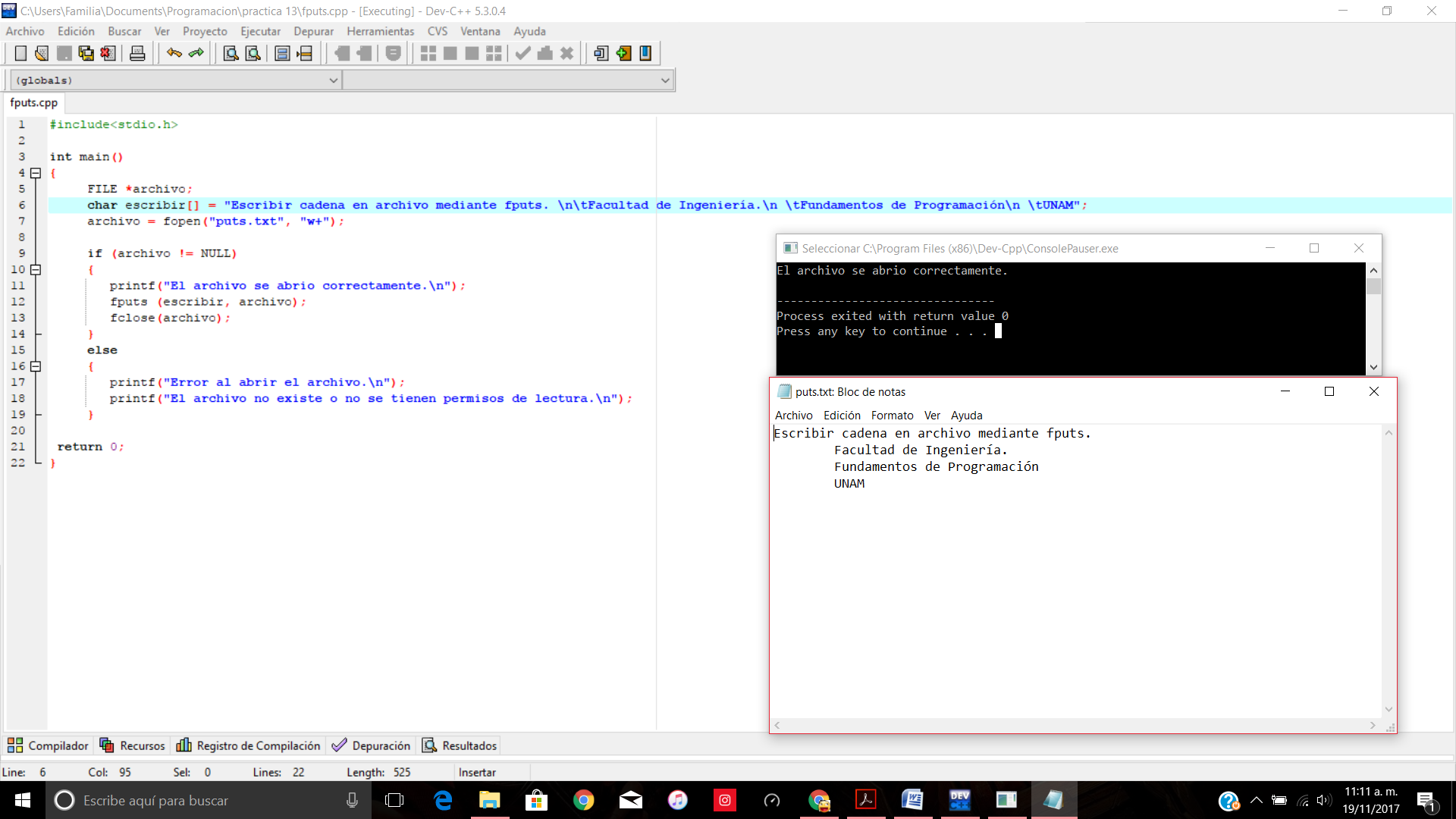
printf("Error al abrir el archivo.\n");

printf("El archivo no existe o no se tienen permisos de lectura.\n");

}

return 0;

}



1. Programa que utiliza la función fgets

#include<stdio.h>

int main()

{

FILE \*archivo;

char caracteres[50];

archivo = fopen("gets.txt", "r");

if (archivo != NULL)

{

printf("El archivo se abrio correctamente.");

printf("\nContenido del archivo:\n");

while (feof(archivo) == 0)

{

fgets (caracteres, 50, archivo);

printf("%s", caracteres);

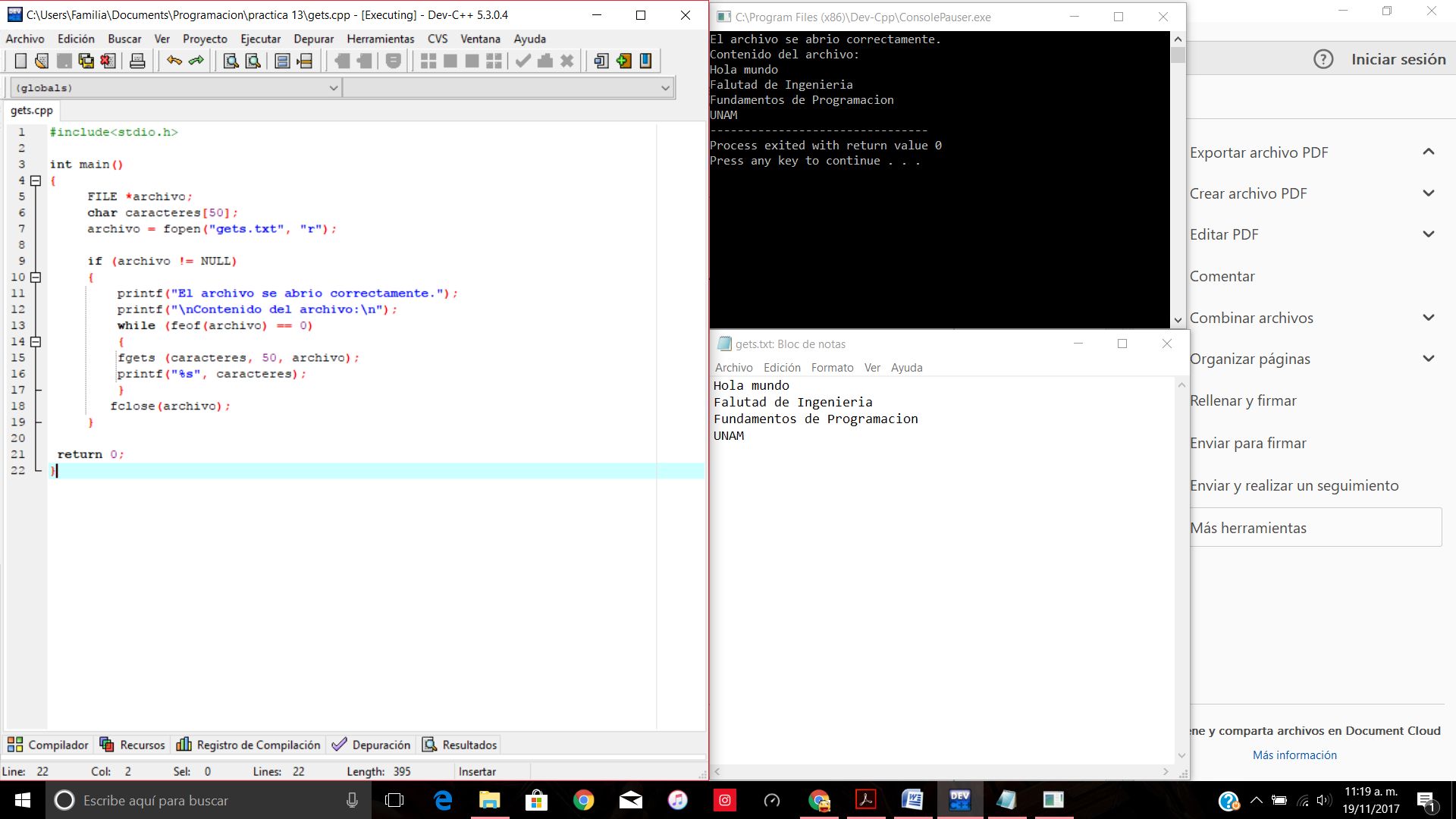
}

fclose(archivo);

}

return 0;

}



1. Programa que usa la funcion fscanf

#include<stdio.h>

int main()

{

FILE \*archivo;

char caracteres[50];

archivo = fopen("fscanf.txt", "r");

if (archivo != NULL)

{

while (feof(archivo)==0)

{

fscanf(archivo, "%s", caracteres);

printf("%s\n", caracteres);

}

fclose(archivo);

}

else

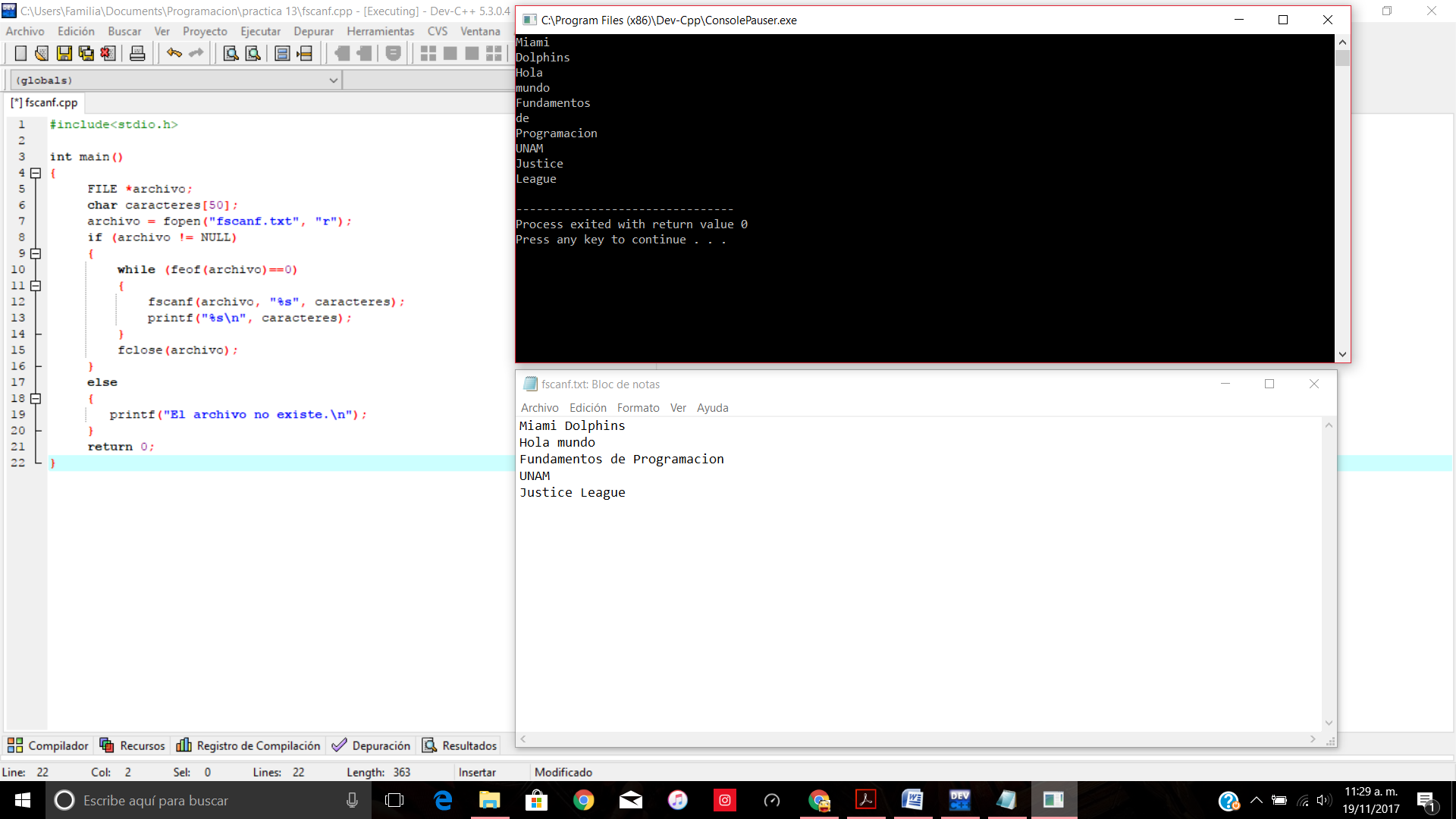
{

printf("El archivo no existe.\n");

}

return 0;

}



1. Programa que utiliza la función fprintf

#include<stdio.h>

int main()

{

FILE \*archivo;

char escribir[] = "Escribir cadena en archivo mediante fprinft. \nFacultad de Ingeniería. \nMiami Dolphins \nBatman\n";

archivo = fopen("fprintf.txt", "r+");

if (archivo != NULL)

{

fprintf(archivo, escribir);

fprintf(archivo, "%s", "UNAM\n");

fclose(archivo);

}

else

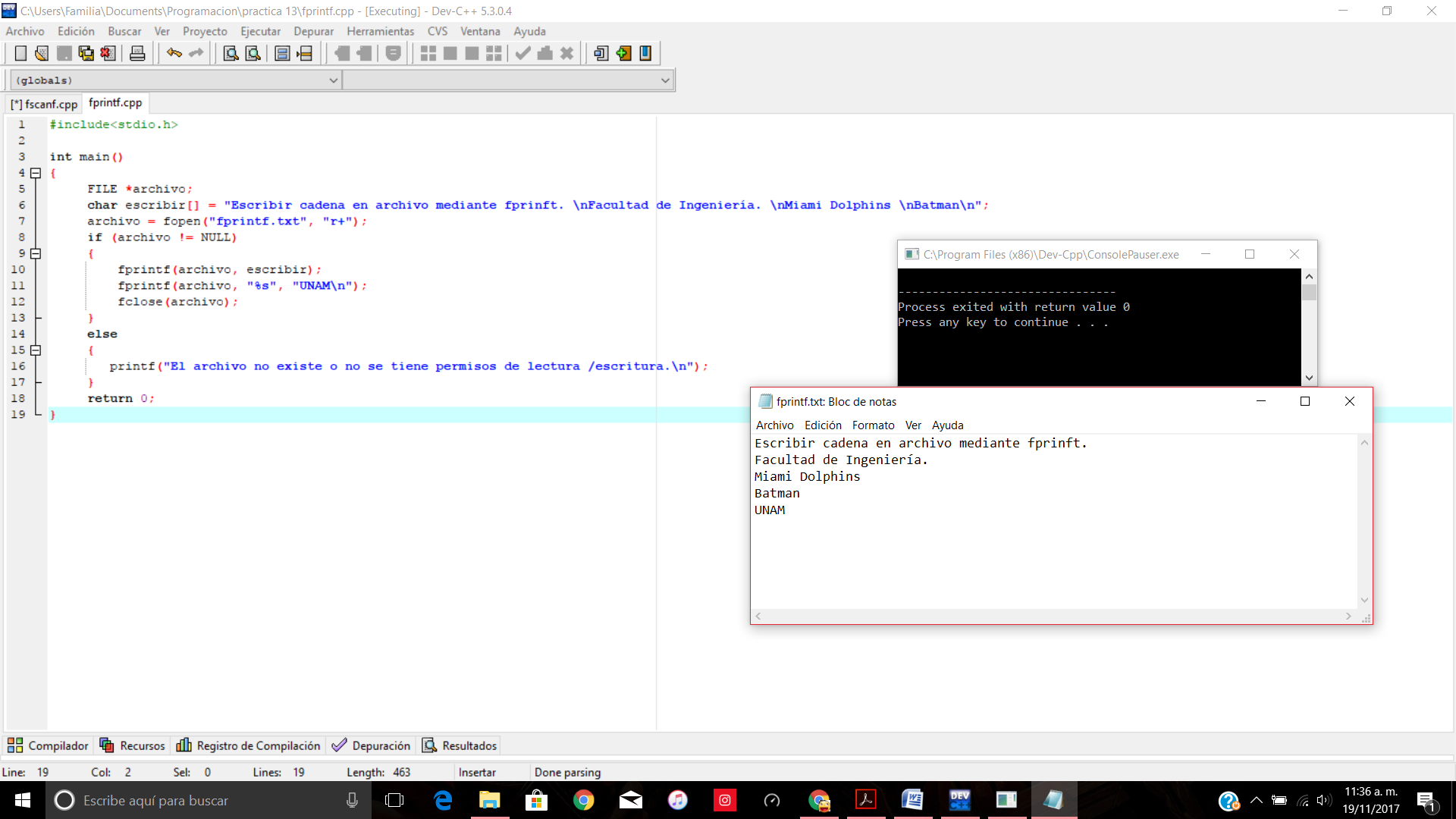
{

printf("El archivo no existe o no se tiene permisos de lectura /escritura.\n");

}

return 0;

}



Conlusiones:

Gracias a esta práctica entendimos mejor lo que era un archivo y la funcionalidad de este, también nos dimos cuenta que los archivos son buenos para almacenar o ingresar información, aparte vimos que las funciones fputs y fgets tienen la misma funcionalidad que fscanf y fprintf.