

Laboratorios de computación salas A y B

Profesor:

Marco Antonio Martínez Quitana

Asignatura:

Principios de Programación

Grupo:

3

No de Práctica(s):

Práctica 12

Integrante(s):

Daniela Cano Ramírez

*No. de Equipo de
cómputo empleado:*

No aplica

No. de Lista o Brigada:

5

Semestre:

Primer Semestre

Fecha de entrega:

Lunes 18 de Enero del 2021

Observaciones:

CALIFICACIÓN: _____

Guía práctica de estudio 12: Arreglos unidimensionales y multidimensionales

Objetivos:

Elaborar programas en C donde la solución del problema se divida en funciones. Distinguir lo que es el prototipo o firma de una función y la implementación de ella, así como manipular parámetros tanto en la función principal como en otras.

Introducción:

Para el desarrollo de tareas dentro de un programa se hace uso de funciones que cumplen una tarea determinada; las funciones principales que se conocen en el mundo de la programación son las *funciones de biblioteca*, que como lo dice su nombre se encuentran predefinidas (scanf, printf, etc).

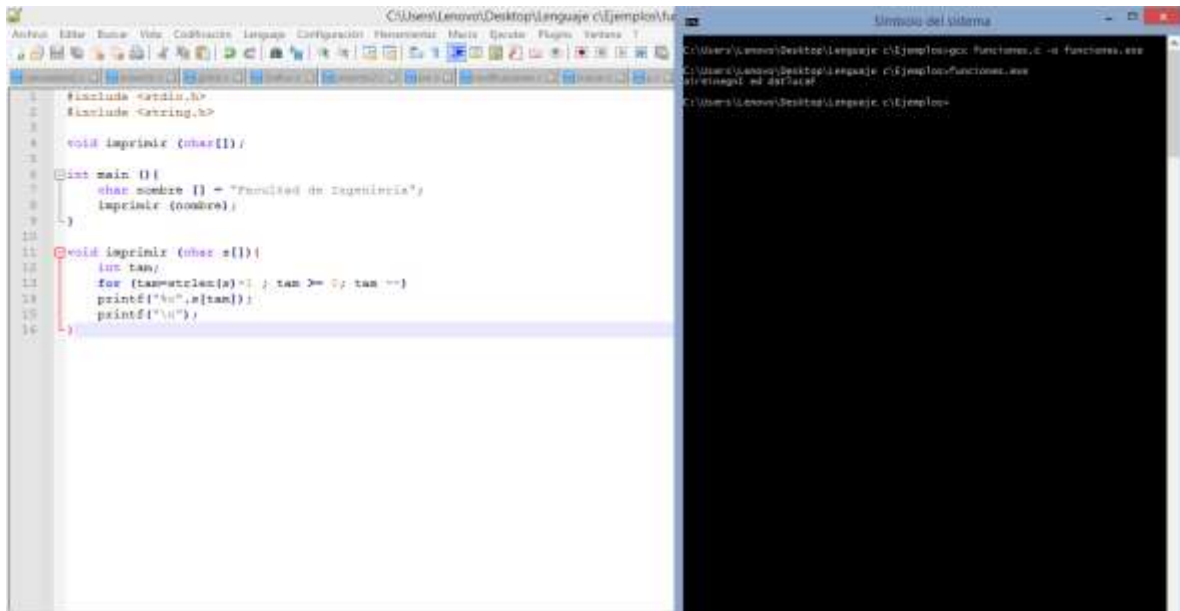
Cuando los programas se vuelven más complejos estas funciones se vuelven un tanto básicas por lo que el programador puede establecer y definir sus propias funciones con el objetivo de que el programa cumpla la funcionalidad, son conocidas como *funciones de usuario*. En trabajos de programación grupales o de gran tamaño se suelen utilizar para fragmentar el proyecto permitiendo un manejo eficiente de los datos.

Actividades:

- Implementar en un programa en C la solución de un problema dividido en funciones.
- Elaborar un programa en C que maneje argumentos en la función principal.
- En un programa en C, manejar variables y funciones estáticas.

Capturas de los códigos ejemplificados en el desarrollo de la práctica:

Código y símbolo del sistema ejemplo 1 Funciones



```
1 #include <stdio.h>
2 #include <string.h>
3
4 void imprimir (char s[]);
5
6 int main () {
7     char nombre [] = "Facultad de Ingeniería";
8     imprimir (nombre);
9 }
10
11 void imprimir (char s[]) {
12     int tam;
13     for (tam=strlen(s)-1; tam >= 0; tam--)
14         printf("%c", s[tam]);
15     printf("\n");
16 }
```

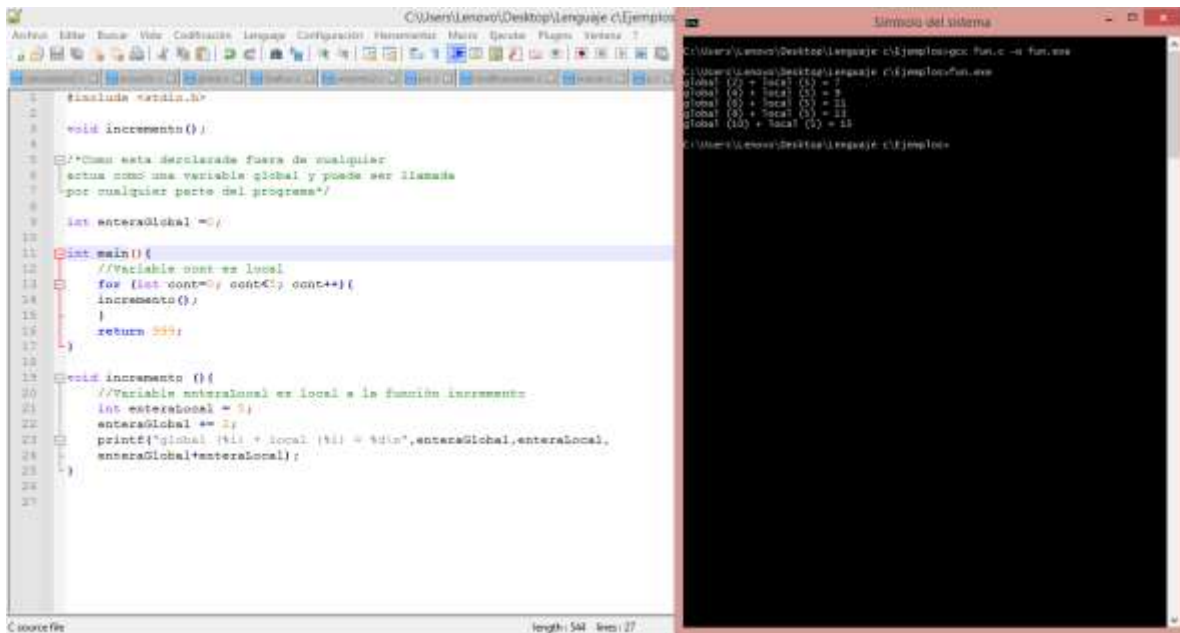
Simbolo del sistema

```
C:\Users\Lenovo\Desktop\language c\ejemplo\funciones.c -o funciones.exe
C:\Users\Lenovo\Desktop\language c\ejemplo\funciones.exe
Facultad de Ingeniería
```

Este programa lo que hace es que la frase contenida en la función principal es reescrita (en sentido contrario) por la función imprimir que se estableció al final del código.

En el caso de este código la función imprimir está declarada de forma global por lo que puede ser llamada desde cualquier parte del programa.

Código y símbolo del sistema ejemplo 2 Ámbito de la variable



The image shows two windows side-by-side. The left window is a code editor titled 'C:\Users\Lenovo\Desktop\lenguaje c\Ejemplo2' containing C code. The right window is a terminal titled 'Símbolo del sistema' showing the output of the program.

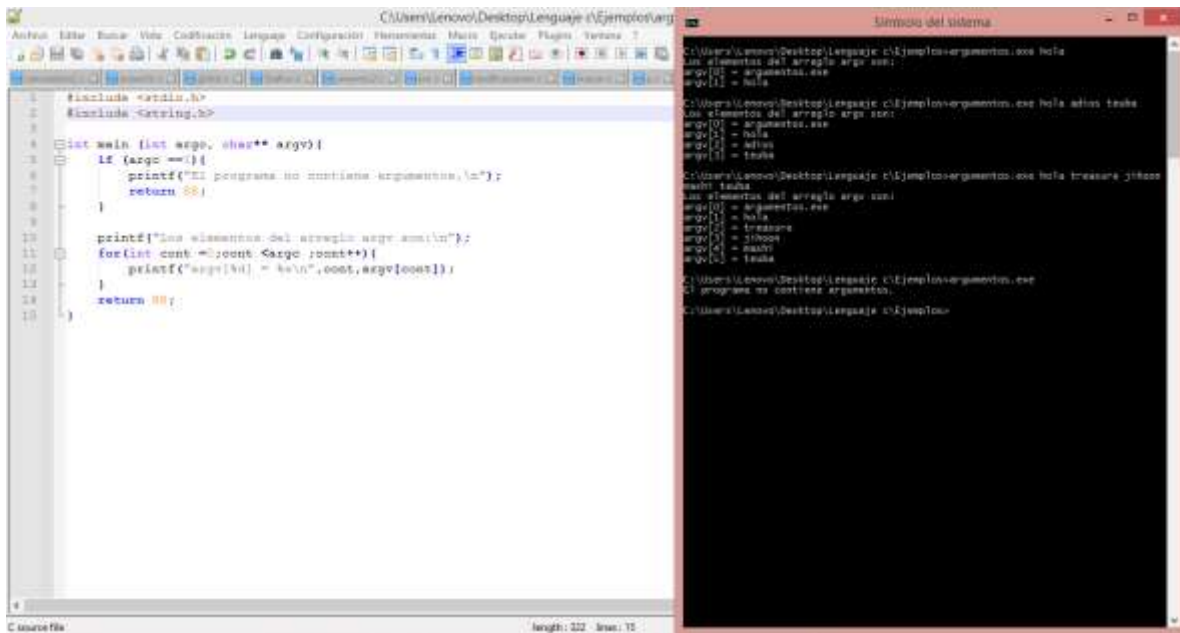
```
1 #include <stdio.h>
2
3 void incremento();
4
5 /*Como esta declarada fuera de cualquier
6  *funcion como una variable global y puede ser llamada
7  *por cualquier parte del programa*/
8
9 int enteraGlobal = 0;
10
11 int main() {
12     //Variable const es local
13     for (int cont=0; cont<5; cont++){
14         incremento();
15     }
16     return 0;
17 }
18
19 void incremento () {
20     //Variable enteraLocal es local a la función incremento
21     int enteraLocal = 5;
22     enteraGlobal += 2;
23     printf("Global (%i) + Local (%i) = %d\n", enteraGlobal, enteraLocal,
24           enteraGlobal+enteraLocal);
25 }
26
27
```

```
C:\Users\Lenovo\Desktop\lenguaje c\Ejemplo2>gcc fun.c -o fun.exe
C:\Users\Lenovo\Desktop\lenguaje c\Ejemplo2>fun.exe
Global (0) + Local (5) = 5
Global (2) + Local (5) = 7
Global (4) + Local (5) = 9
Global (6) + Local (5) = 11
Global (8) + Local (5) = 13
Global (10) + Local (5) = 15
C:\Users\Lenovo\Desktop\lenguaje c\Ejemplo2>
```

Para este programa el contador se declaró en la función principal y posteriormente se hizo llamado a la función incremento. Esta función contiene los calculos pero todo el proceso se realizó en la función incremento que contenía a la variable local de entero “5”.

En pantalla se muestran las primeras 5 sumatorias de números pares más la variable local que es “5”.

Código y símbolo del sistema ejemplo 3 Argumentos función main



The image shows a screenshot of a C++ development environment. On the left, the source code for a program named 'ejemplo3.cpp' is displayed. The code includes `<string.h>` and defines a `main` function that takes an array of strings `argv`. It checks if `argc` is 1, and if so, prints 'El programa no contiene argumentos.' and returns 88. Otherwise, it prints 'Los elementos del arreglo argv son:' and then iterates through `argv` from index 1 to `argc-1`, printing each element. It returns 88 at the end. On the right, a 'Símbolo del sistema' (System Symbol) window shows the output of the program. It displays the path to the executable, the command to run it, and the output of the program for three different inputs: no arguments, one argument 'adios tesha', and three arguments 'hola tesha', 'tesha', and 'tesha'.

```
#include <string.h>
#include <string.h>

int main (int argc, char** argv){
    if (argc == 1){
        printf("El programa no contiene argumentos.\n");
        return 88;
    }

    printf("Los elementos del arreglo argv son:\n");
    for(int cont = 0; cont < argc; cont++){
        printf("argv[%d] = %s\n", cont, argv[cont]);
    }
    return 88;
}
```

C:\Users\Lenovo\Desktop\lenguaje c\ejemplo3-argumentos.exe hola
Los elementos del arreglo argv son:
argv[0] = argumento.exe
argv[1] = hola

C:\Users\Lenovo\Desktop\lenguaje c\ejemplo3-argumentos.exe hola adios tesha
Los elementos del arreglo argv son:
argv[0] = argumento.exe
argv[1] = hola
argv[2] = adios
argv[3] = tesha

C:\Users\Lenovo\Desktop\lenguaje c\ejemplo3-argumentos.exe hola tesha tesha tesha
Los elementos del arreglo argv son:
argv[0] = argumento.exe
argv[1] = hola
argv[2] = tesha
argv[3] = tesha
argv[4] = tesha

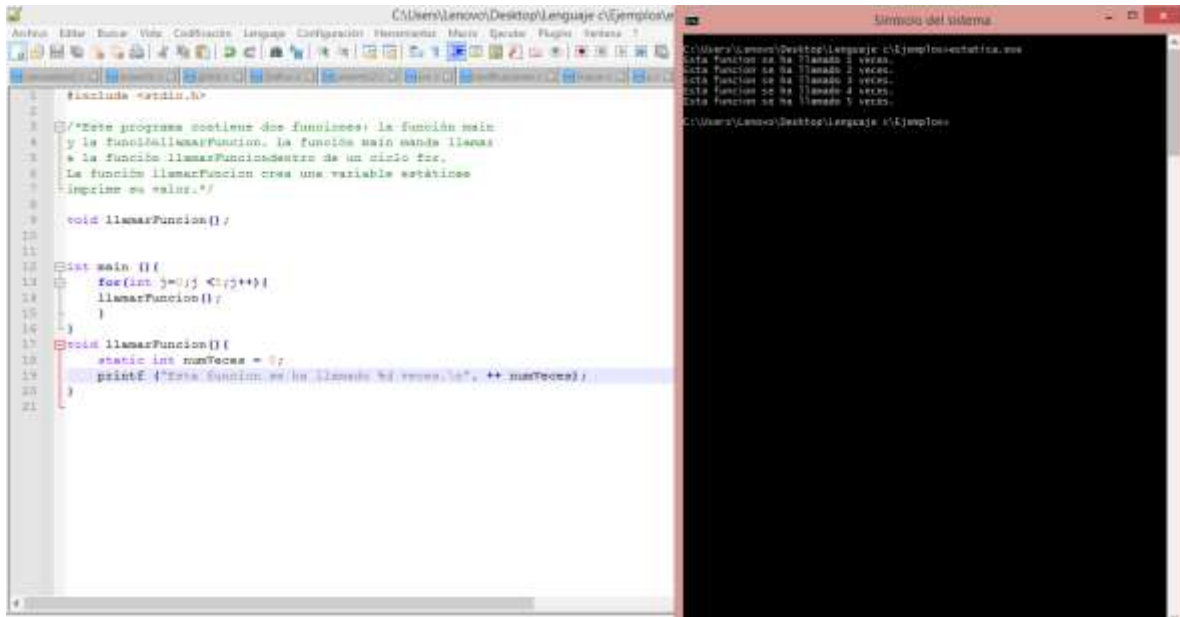
C:\Users\Lenovo\Desktop\lenguaje c\ejemplo3-argumentos.exe
El programa no contiene argumentos.
C:\Users\Lenovo\Desktop\lenguaje c\ejemplo3-

En el caso de este programa lo que se observa es que al darle argumentos a la función principal se le deben meter los datos a ocupar por el programa

Argc - Actúa como un contador (de la cantidad de parámetros ingresados en la función).

Si no se ingresa ningún parámetro se imprime la línea "El programa no tiene argumentos".

Código y símbolo del sistema ejemplo 4 Variable estática



The image shows a C++ IDE with two windows. The left window displays the source code for a program that demonstrates static variables. The right window shows the output of the program when executed in a command prompt.

```
1 #include <stdio.h>
2
3 /*Este programa contiene dos funciones: la función main
4 y la función llamarFuncion. La función main llama
5 a la función llamarFuncion dentro de un ciclo for.
6 La función llamarFuncion crea una variable estática
7 y imprime su valor.*/
8
9 void llamarFuncion()
10
11
12 int main ()
13 {
14     for(int i=0; i<5;i++){
15         llamarFuncion();
16     }
17 }
18
19 void llamarFuncion() {
20     static int numVeces = 0;
21     printf ("Esta función se ha llamado %d veces.\n", ++ numVeces);
22 }
```

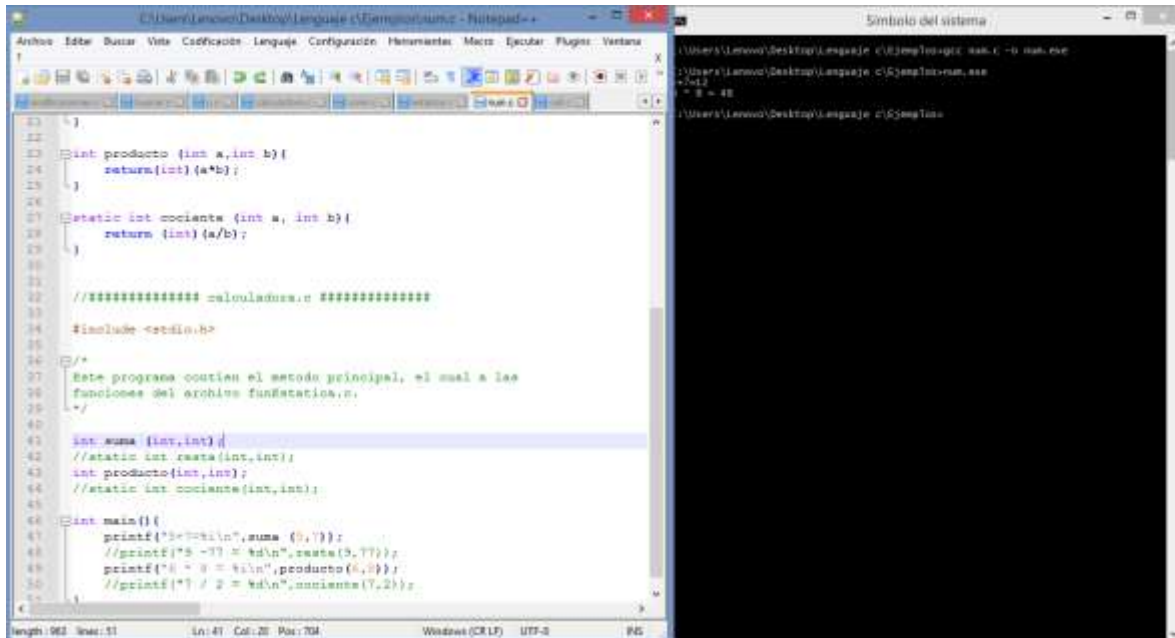
Simbolo del sistema

```
C:\Users\Lenovo\Desktop\lenguaje c\ejemplos>ejemplo-estatica.exe
Esta función se ha llamado 1 veces.
Esta función se ha llamado 2 veces.
Esta función se ha llamado 3 veces.
Esta función se ha llamado 4 veces.
Esta función se ha llamado 5 veces.
C:\Users\Lenovo\Desktop\lenguaje c\ejemplos>
```

En la función principal tenemos un bucle for y después se llama a la función “llamarFunción” con la cual se imprime la línea de texto que indica la cantidad de veces que se ha llamado a la variable numVeces.

Como nuestro bucle es de 5 la cantidad de veces que se hace el llamado de la función es en 5 ocasiones.

Código y símbolo del sistema ejemplo 5 Variable estática 2



The screenshot shows a C program in Notepad++ and its execution in a command prompt. The program defines a static variable `cociente` and a function `cociente` that returns `a/b`. It also defines a function `suma` that calls `suma`, `resta`, `producto`, and `cociente`. The `main` function prints the results of these operations.

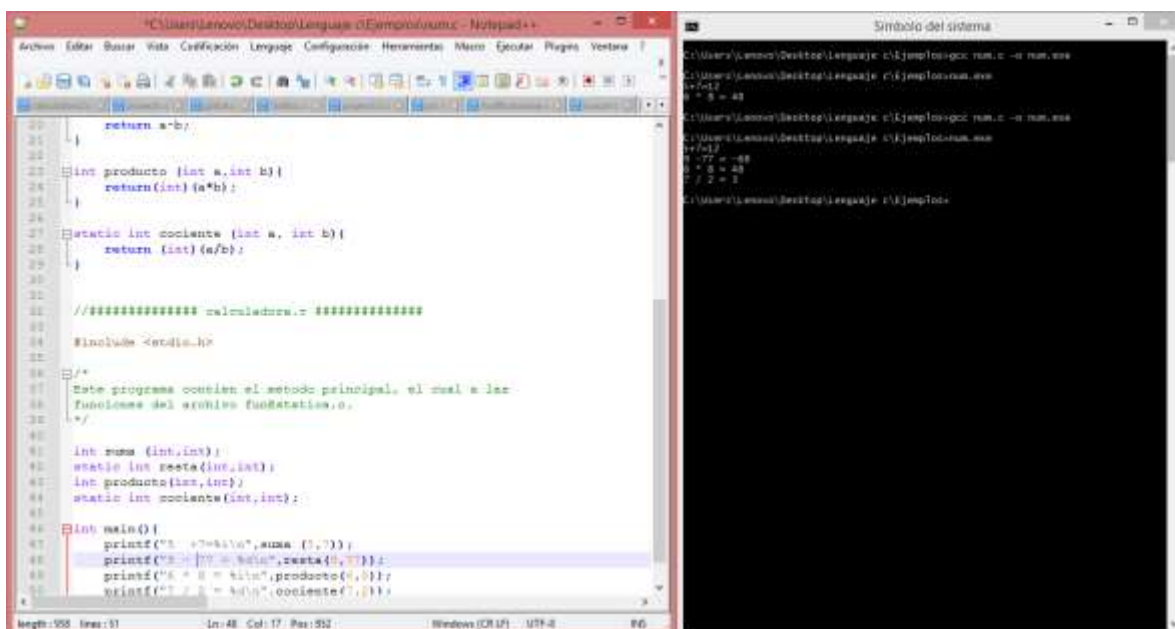
```
1 //***** calculadora.c *****/
2
3 #include <stdio.h>
4
5 /*
6  * Este programa contiene el método principal, el cual a las
7  * funciones del archivo funEstatica.c.
8  */
9
10 int suma (int,int);
11 static int resta (int,int);
12 int producto (int,int);
13 static int cociente (int,int);
14
15 int main() {
16     printf("5 + 7 = %d\n", suma (5,7));
17     printf("5 - 7 = %d\n", resta (5,7));
18     printf("6 * 8 = %d\n", producto (6,8));
19     printf("7 / 2 = %d\n", cociente (7,2));
20 }
```

The command prompt shows the execution of the program, displaying the results of the operations:

```
C:\Users\Lenovo\Desktop\Language c\Ejemplo5> gcc sum.c -o sum.exe
C:\Users\Lenovo\Desktop\Language c\Ejemplo5> ./sum.exe
5 + 7 = 12
5 - 7 = -2
6 * 8 = 48
7 / 2 = 3
```

Tal como dice la práctica si ejecutamos el programa con las variables declaradas únicamente de forma global podemos acceder a ambas operaciones.

Cuando modificamos el código es posible ejecutar las 4 operaciones



The screenshot shows a modified C program in Notepad++ and its execution in a command prompt. The program now defines a static variable `suma` and a function `suma` that calls `suma`, `resta`, `producto`, and `cociente`. The `main` function prints the results of these operations.

```
20     return a*b;
21
22 int producto (int a,int b){
23     return (int) (a*b);
24 }
25
26 static int cociente (int a, int b){
27     return (int) (a/b);
28 }
29
30
31 //***** calculadora.c *****/
32
33 #include <stdio.h>
34
35 /*
36  * Este programa contiene el método principal, el cual a las
37  * funciones del archivo funEstatica.c.
38  */
39
40 int suma (int,int);
41 static int resta (int,int);
42 int producto (int,int);
43 static int cociente (int,int);
44
45 int main() {
46     printf("5 + 7 = %d\n", suma (5,7));
47     printf("5 - 7 = %d\n", resta (5,7));
48     printf("6 * 8 = %d\n", producto (6,8));
49     printf("7 / 2 = %d\n", cociente (7,2));
50 }
```

The command prompt shows the execution of the program, displaying the results of the operations:

```
C:\Users\Lenovo\Desktop\Language c\Ejemplo5> gcc sum.c -o sum.exe
C:\Users\Lenovo\Desktop\Language c\Ejemplo5> ./sum.exe
5 + 7 = 12
5 - 7 = -2
6 * 8 = 48
7 / 2 = 3
```

Referencias

Funciones en C. (s. f.). Recuperado 18 de enero de 2021, de

<https://www.mheducation.es/bcv/guide/capitulo/8448148681.pdf>

Laboratorio de Computación Salas A y B. Facultad de Ingeniería (2018, abril 6). Manual de prácticas Fundamento de Programación. Recuperado 18 de Enero de 2021, de

<http://lcp02.fi-b.unam.mx/>