|  |
| --- |
| Laboratorio de Computación  Salas A y B |
| |  |  | | --- | --- | | Profesor: | Claudia Rodríguez Espino | | Asignatura: | Fundamentos de Programación | | Grupo: | 1104 | | No de Práctica(s): | 12 | | Integrante(s): | Téllez Torres Lorena Alejandra | |  |  | |  |  | |  |  | | No. de Equipo de cómputo empleado: | 51 | | Semestre: | 1ª | | Fecha de entrega: |  | | Observaciones: |  | |  |  | |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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.

**Desarrollo**

Como ya se mencionó, un programa en lenguaje C consiste en una o más funciones. C permite tener dentro de un archivo fuente varias funciones, esto con el fin de dividir las tareas y que sea más fácil la depuración, la mejora y el entendimiento del código.

En lenguaje C la función principal se llama main. Cuando se ordena la ejecución del programa, se inicia con la ejecución de las instrucciones que se encuentran dentro de la función main, y ésta puede llamar a ejecutar otras funciones, que a su vez éstas pueden llamar a ejecutar a otras funciones, y así sucesivamente.

**Funciones**

La sintaxis básica para definir una función es la siguiente:

*valorRetorno nombre (parámetros){*

*// bloque de código de la función*

*}*

El nombre de la función se refiere al identificador con el cual se ejecutará la función; se debe seguir la notación de camello.

Una función puede recibir parámetros de entrada, los cuales son datos de entrada con los que trabajará la función, dichos parámetros se deben definir dentro de los paréntesis de la función, separados por comas e indicando su tipo de dato, de la siguiente forma:

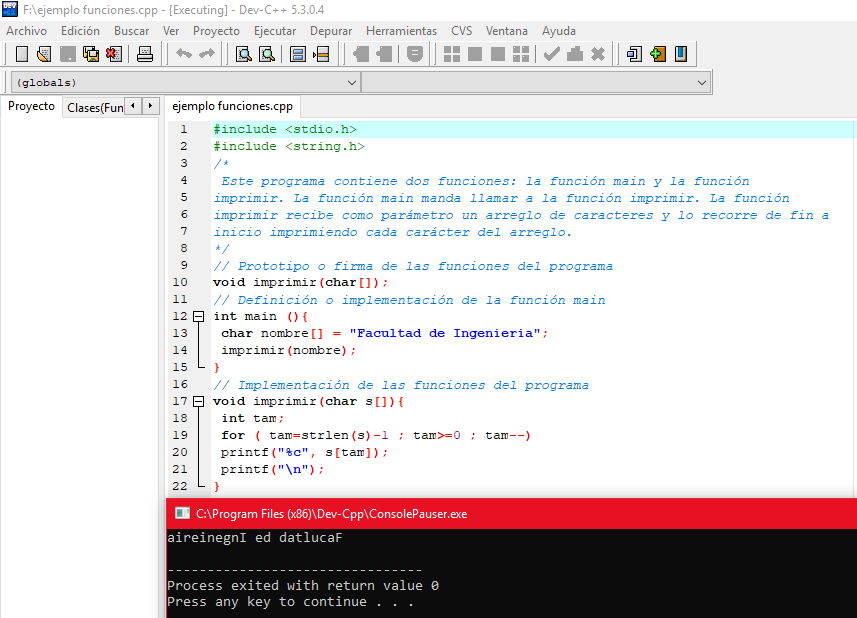
*(tipoDato nom1, tipoDato nom2, tipoDato nom3…)*

El compilador C revisa que las funciones estén definidas o declaradas antes de ser

invocadas. Por lo que una buena práctica es declarar todas las funciones al inicio del programa. Una declaración, prototipo o firma de una función tiene la siguiente sintaxis:

*valorRetorno nombre (parámetros);*

**Código (funciones)**

****

**Argumentos para la función main**

Como se mencionó anteriormente, la firma de una función está compuesta por tres elementos: el nombre de la función, los parámetros que recibe la función y el valor de retorno de la función.

La función main también puede recibir parámetros. Debido a que la función main es la primera que se ejecuta en un programa, los parámetros de la función hay que enviarlos al ejecutar el programa. La firma completa de la función main es:

*int main (int argc, char \*\* argv);*

La función main puede recibir como parámetro de entrada un arreglo de cadenas al

ejecutar el programa. La longitud del arreglo se guarda en el primer parámetro *(argument counter)* y el arreglo de cadenas se guarda en el segundo parámetro *(argument vector).*

Para enviar parámetros, el programa se debe ejecutar de la siguiente manera:

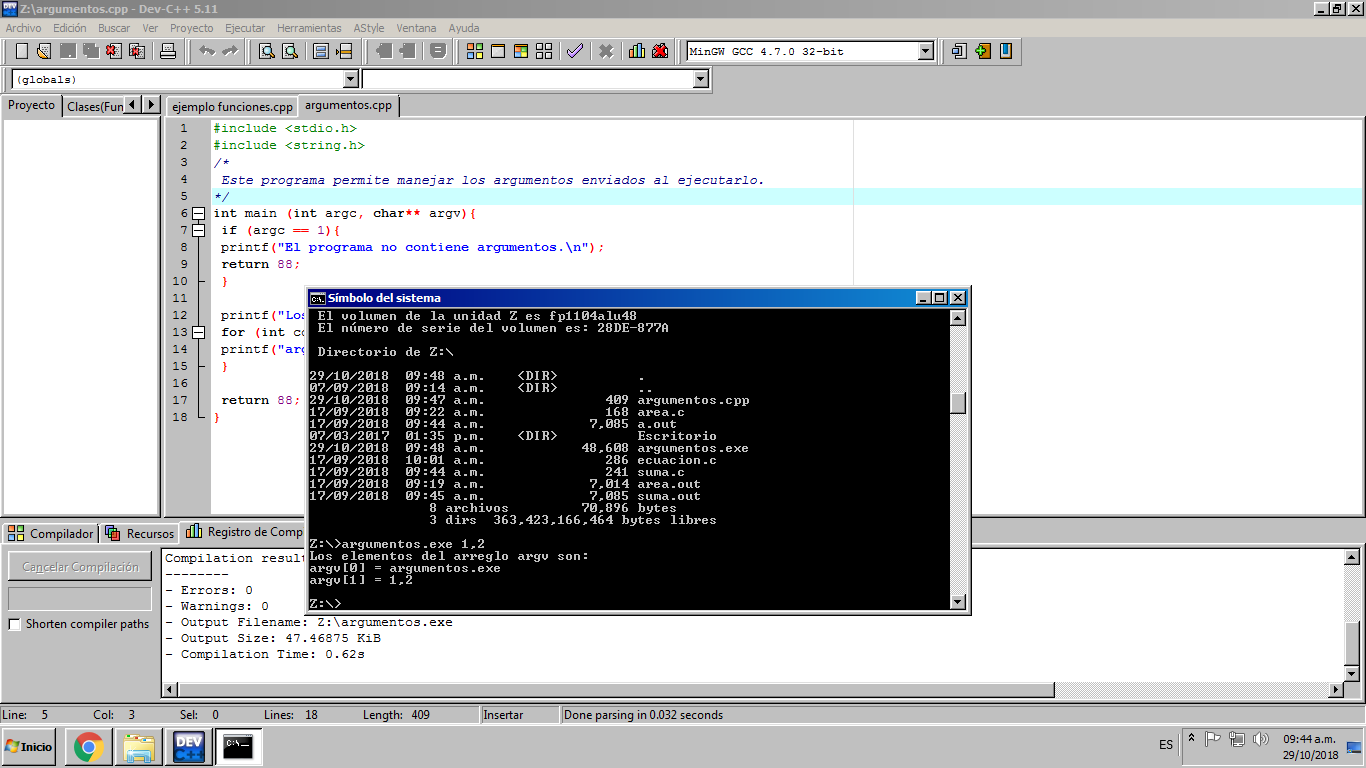
* En plataforma Linux/Unix

*./nombrePrograma arg1 arg2 arg3 ...*

*En plataforma Windows*

*nombrePrograma.exe arg1 arg2 arg3 ...*

Esto es, el nombre del programa seguido de los argumentos de entrada. Estos argumentos son leídos como cadenas de caracteres dentro del *argument vector*, donde en la posición 0 se encuentra el nombre del programa, en la posición 1 el primer argumento, en la posición 2 el segundo argumento y así sucesivamente.



**Estático**

Lenguaje C permite definir elementos estáticos. La sintaxis para declarar elementos

estáticos es la siguiente:

*static tipoDato nombre;*

*static valorRetorno nombre(parámetros);*

Es decir, tanto a la declaración de una variable como a la firma de una función solo se le agrega la palabra reservada static al inicio de las mismas.

El atributo static en una variable hace que ésta permanezca en memoria desde su creación y durante toda la ejecución del programa, lo que quiere decir que su valor se mantendrá hasta que el programa llegue a su fin.

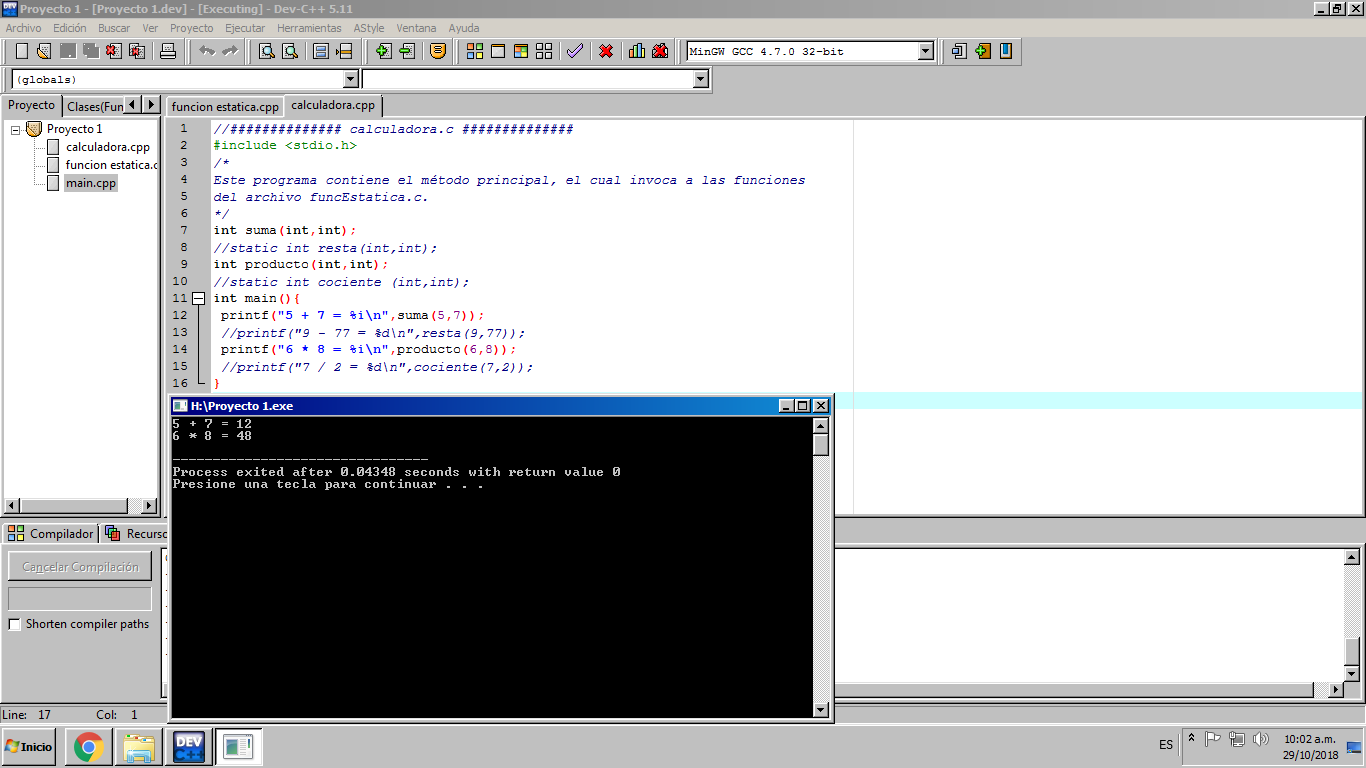
El atributo static en una función hace que esa función sea accesible solo dentro del mismo archivo, lo que impide que fuera de la unidad de compilación se pueda acceder a la función.

**Código (variable estática)**

Una vez declarada una variable estática, esta permanece en memoria a lo largo de la ejecución del programa, por lo tanto, la segunda vez que se llama a la función ya no se vuelve a crear la variable, si no que se utiliza la que está en la memoria y por eso conserva su valor.

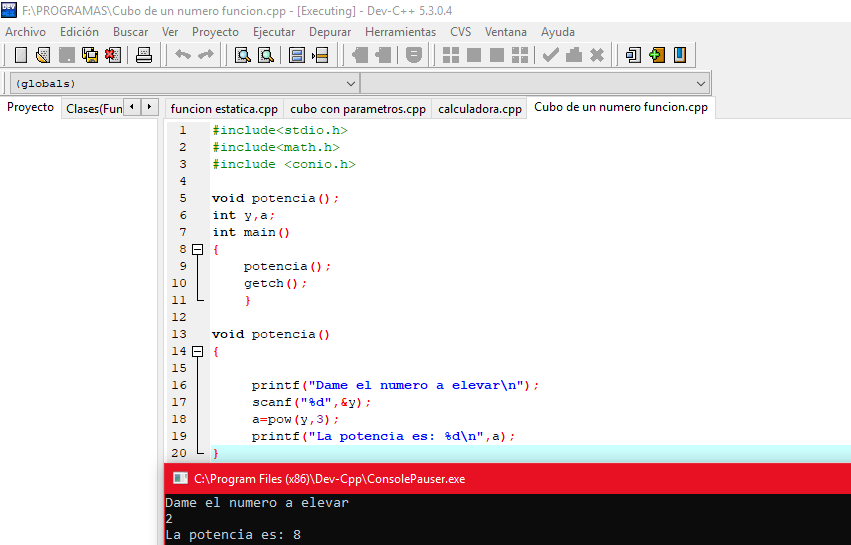
**Código (función estática)**

Cuando se compilan los dos archivos al mismo tiempo *(gcc funcEstatica.c calculadora.c –o exe)*, las funciones suma y producto son accesibles desde el archivo calculadora y, por tanto, se genera el código ejecutable. Si se quitan los comentarios y se intenta compilar los archivos se enviará un error, debido a que las funciones son estáticas y no pueden ser accedidas fuera del archivo *funcEstaticas.c.*

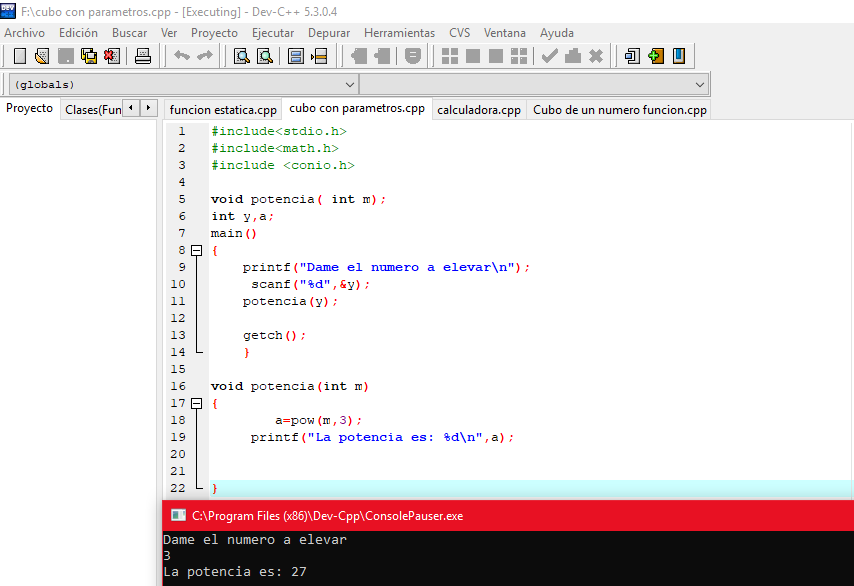


**Ejercicios**

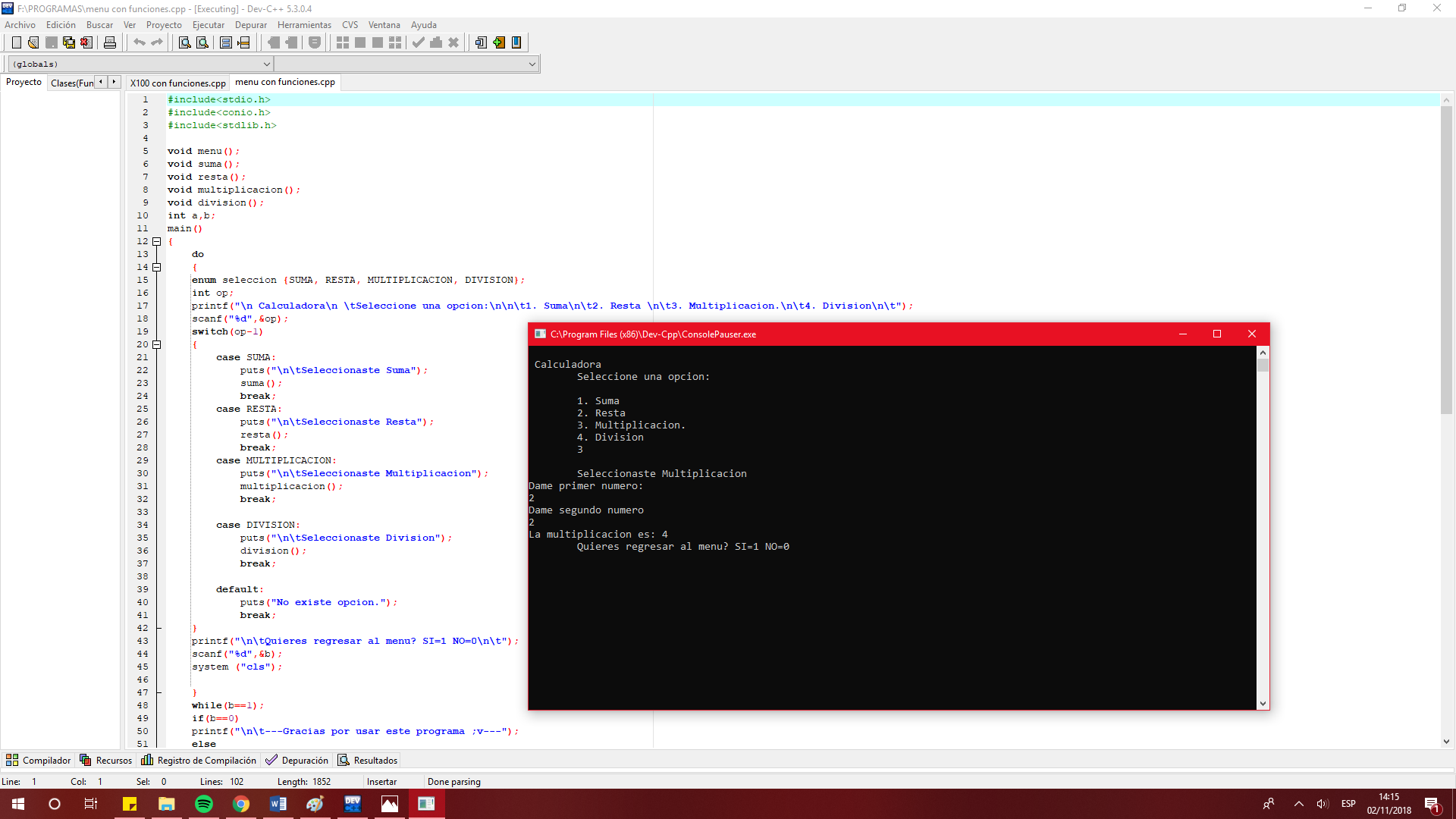
1. **Cubo de un número (función sin parámetros)**

****

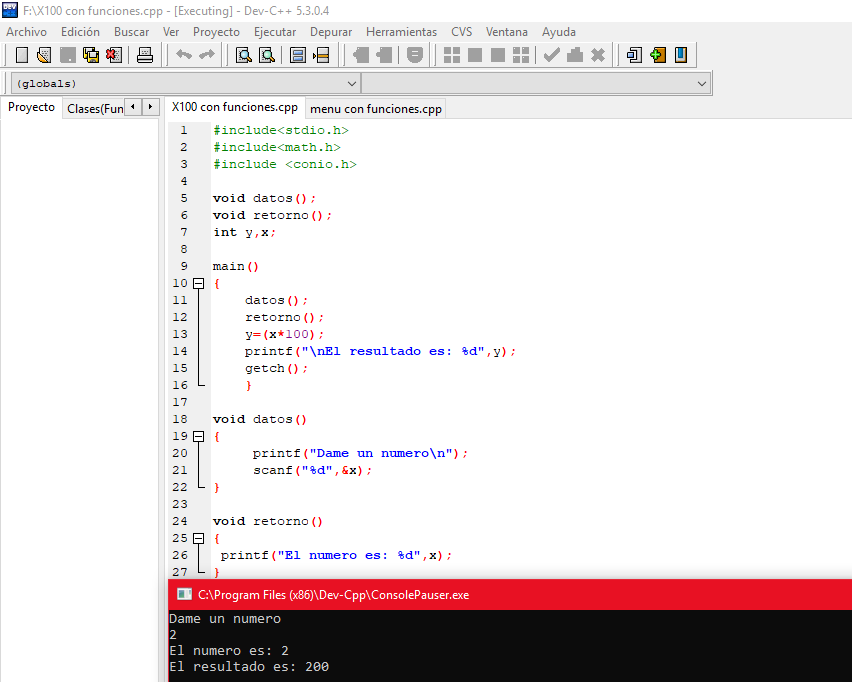
1. **Cubo de un número (función con parámetros)**

****

1. **Calculadora**



1. **Función que llama a otra función**

****

**Conclusiones**

Me gustó mucho la práctica porque no sabía de la existencia de los “proyectos”, son muy útiles para agrupar programas en un solo archivo. También me gustó la parte de funciones ya que se me dificultaban mucho las funciones con parámetros, pero creo que todo se basa en práctica ya que tenemos las bases para crear un programa sólo es comprenderlo, y aterrizar la idea que tienes como solución. Siempre me ha gustado saber varias formas de hacer los programas y diferentes formas de estructuración de éstos.