



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Ing. Karina García Morales

Asignatura: Fundamentos de la Programación

Grupo: 20

No. de práctica(s): 11

Integrante(s): Martínez Ordoñez Diego Tonatiah

No. de lista o brigada: 30

Semestre: 2023-1

Fecha de entrega: 14 / diciembre / 2022

Observaciones:

CALIFICACIÓN: _____

Práctica 11: Funciones

- Objetivo:

El alumno elaborará 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:

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:

```
tipoValorRetorno 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, 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:

tipoValorRetorno nombre (parámetros);

La firma de una función está compuesta por tres elementos: el tipo del valor de retorno de la función, el nombre de la función y los parámetros que recibe la función; finaliza con punto y coma (;). Los nombres de los parámetros no necesariamente deben ser iguales a los que se encuentran en la definición de la función. Las funciones definidas en el programa no necesariamente deberán ser declaradas; esto dependerá de su ubicación en el Código.

- Códigos en clase:

1. El siguiente programa contiene dos funciones: la función main y la función imprimir. La función main manda llamar a la función imprimir. La función imprimir recibe como parámetro un arreglo de caracteres y lo recorre de fin a inicio imprimiendo cada carácter del arreglo.

```
1 //  
2 // main.c  
3 // exercise1  
4 //  
5 // Created by Martinez Ordenez Diego Tonatiuh on 12/7/22.  
6 // Copyright © 2022 Martinez Ordenez Diego Tonatiuh. All rights reserved.  
7 //  
8 //  
9 #include <stdio.h>  
10 #include <string.h>  
11 // Prototipo o firma de las funciones del programa  
12 void imprimir(char[]);  
13 // Definición o implementación de la función main  
14 void imprimir(char s[]){ int tam;  
15 for (tam=strlen(s)-1; tam>=0; tam--) printf("%c", s[tam]);  
16 printf("\n");  
17 }  
18 int main () {  
19 char nombre[] = "Facultad de Ingeniería"; imprimir(nombre);  
20 }  
21 // Implementación de las funciones del programa  
22  
23
```

Facultad de Ingeniería
Program ended with exit code: 0

```
1 //  
2 // main.c  
3 // exercise1  
4 //  
5 // Created by Martinez Ordenez Diego Tonatiuh on 12/7/22.  
6 // Copyright © 2022 Martinez Ordenez Diego Tonatiuh. All rights reserved.  
7 //  
8 //  
9 #include <stdio.h>  
10 #include <string.h>  
11 // Prototipo o firma de las funciones del programa  
12 void imprimir(char[]);  
13 // Definición o implementación de la función main  
14 void imprimir(char s[])  
15 { int tam;  
16 for (tam=0; tam<strlen(s)-1; tam++) printf("%c", s[tam]);  
17 printf("\n");  
18 }  
19 int main ()  
20 {  
21 char nombre[] = "Facultad de Ingeniería"; imprimir(nombre);  
22 }  
23 // Implementación de las funciones del programa  
24  
25
```

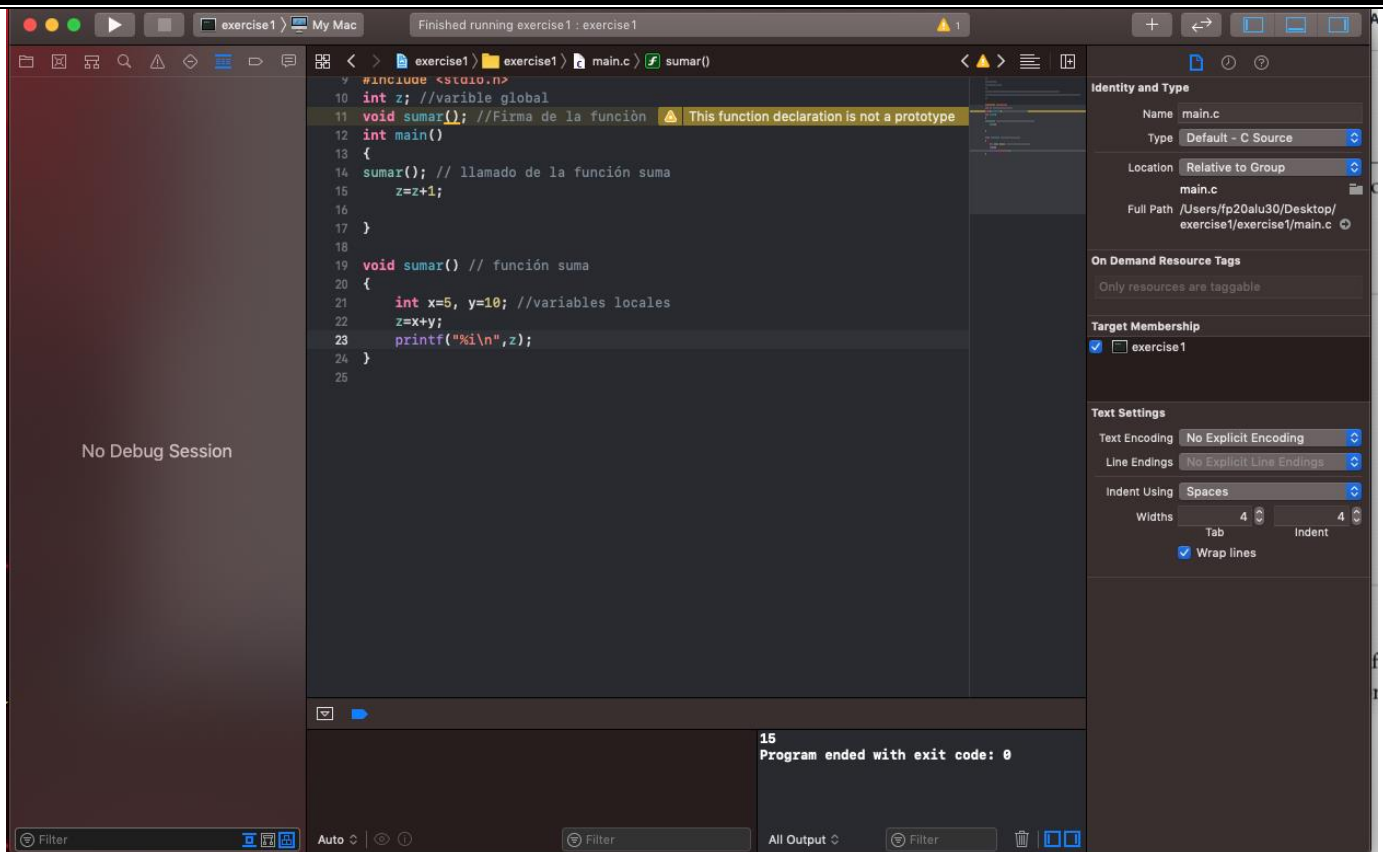
Facultad de Ingeniería
Program ended with exit code: 0

- **Ámbito o alcance de las variables.**

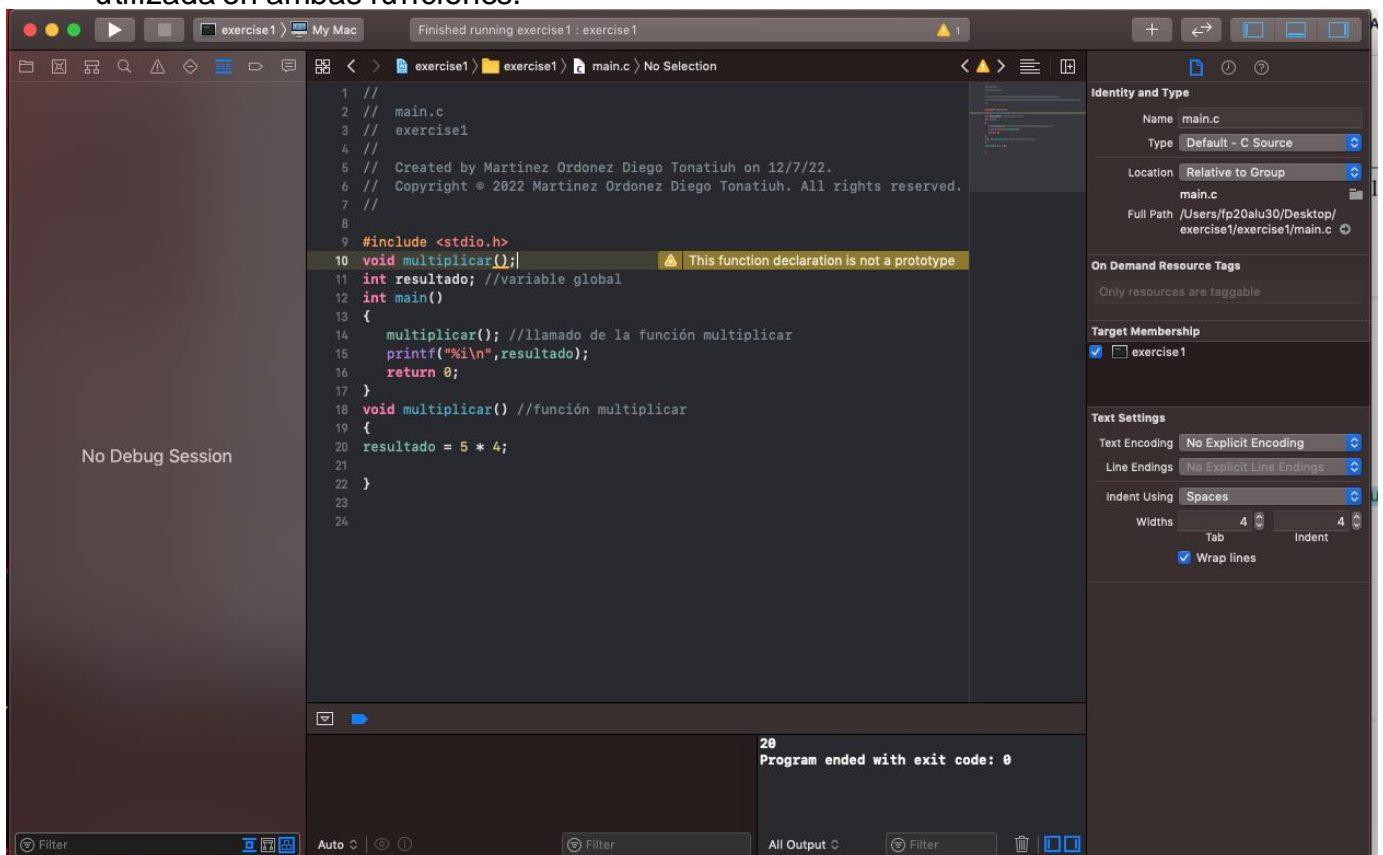
Las variables declaradas dentro de un programa tienen un tiempo de vida que depende de la posición donde se declaren. En C existen dos tipos de variables con base en el lugar donde se declaren: variables locales y variables globales.

- **Código en clase:**

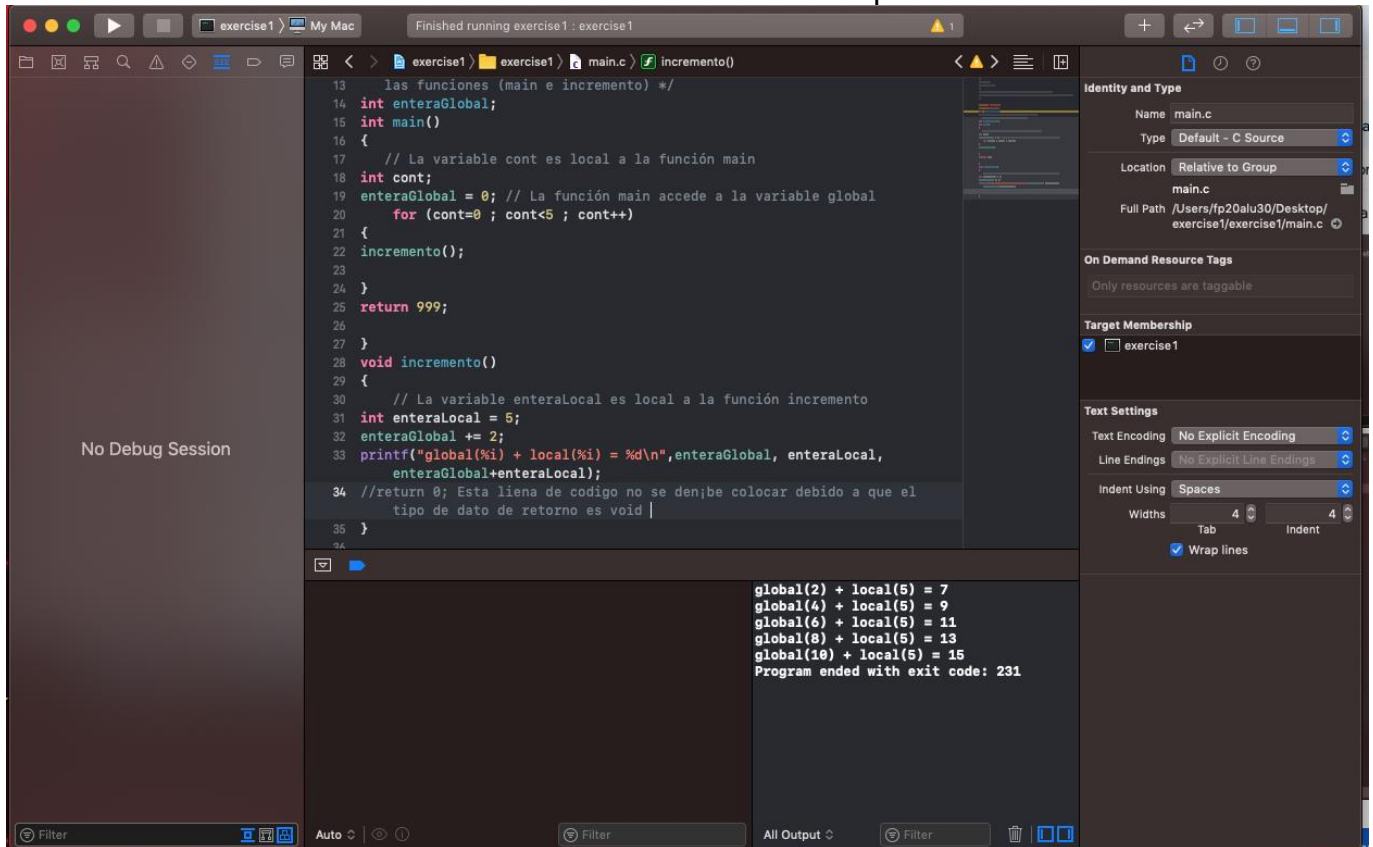
2. El siguiente programa muestra la declaración y uso de variables locales en la función de suma.



3. El siguiente programa muestra la declaración de la variable global resultado, la cual es utilizada en ambas funciones.



4. Este programa contiene dos funciones: la función main y la función incremento. La función main manda llamar a la función incremento dentro de un ciclo for. La función incremento aumenta el valor de la variable enteraGlobal cada vez que es invocada.



```
13 // las funciones (main e incremento) */
14 int enteraGlobal;
15 int main()
16 {
17     // La variable cont es local a la función main
18     int cont;
19     enteraGlobal = 0; // La función main accede a la variable global
20     for (cont=0 ; cont<5 ; cont++)
21     {
22         incremento();
23     }
24     return 999;
25 }
26
27 void incremento()
28 {
29     // La variable enteraLocal es local a la función incremento
30     int enteraLocal = 5;
31     enteraGlobal += 2;
32     printf("global(%i) + local(%i) = %d\n", enteraGlobal, enteraLocal,
33           enteraGlobal+enteraLocal);
34     //return 0; Esta línea de código no se debe colocar debido a que el
35     //tipo de dato de retorno es void
36 }
```

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
Program ended with exit code: 231

- Argumentos para la función main:

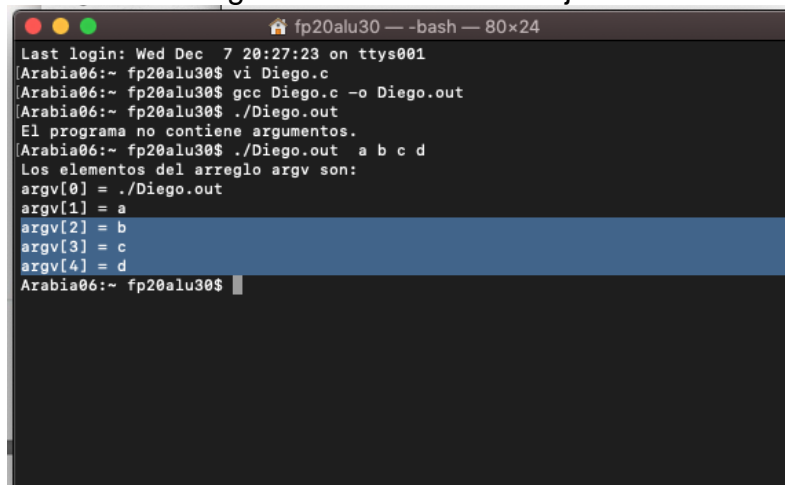
La función main también puede recibir parámetros. Debido a que la función main es la primera que se ejecuta en un programa, los parámetros de la función hay que enviarlos al ejecutar el programa. La firma completa de la función main es:

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

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

- En plataforma Linux/Unix
`./nombrePrograma arg1 arg2 arg3 ...`
- En plataforma Windows
`nombrePrograma.exe arg1 arg2 arg3 ...`
- Códigos en clase:

5. Este programa muestra los argumentos enviados al ejecutarlo



```
Last login: Wed Dec 7 20:27:23 on ttys001
[Arabia06:~ fp20alu30$ vi Diego.c
[Arabia06:~ fp20alu30$ gcc Diego.c -o Diego.out
[Arabia06:~ fp20alu30$ ./Diego.out
El programa no contiene argumentos.
[Arabia06:~ fp20alu30$ ./Diego.out a b c d
Los elementos del arreglo argv son:
argv[0] = ./Diego.out
argv[1] = a
argv[2] = b
argv[3] = c
argv[4] = d
[Arabia06:~ fp20alu30$
```

- Estático:

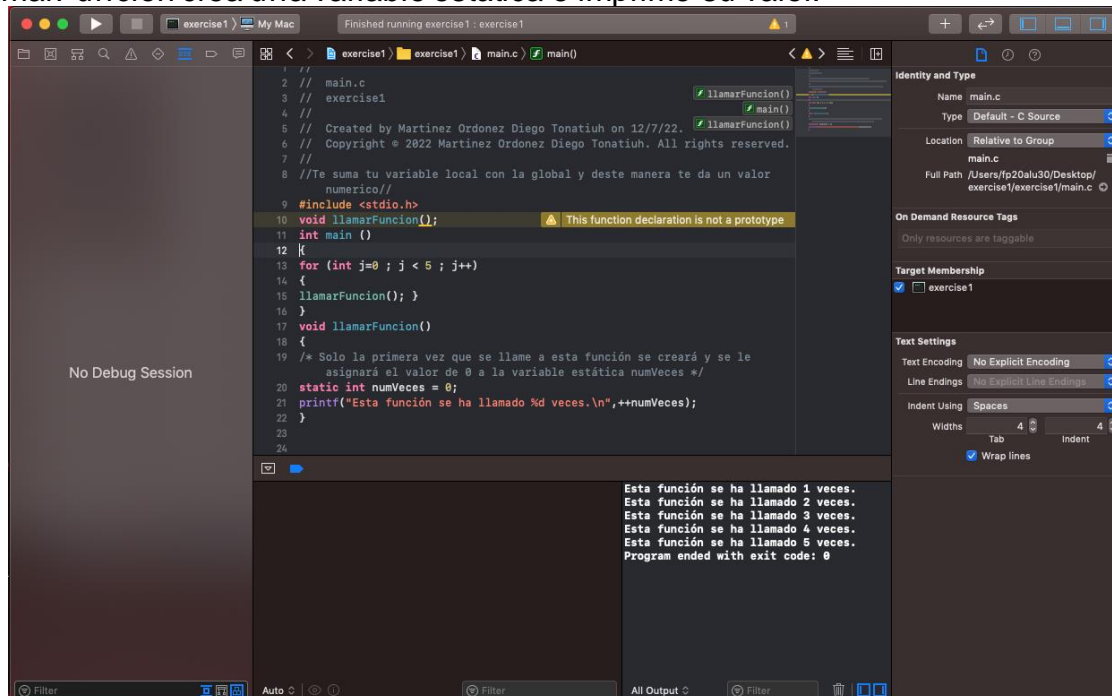
Lenguaje C permite definir elementos estáticos. La sintaxis para declarar elementos estáticos es la siguiente: `static tipoDato nombre;`

`static tipoValorRetorno nombre(parámetros);`

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 en clase:

- Este programa contiene dos funciones: la función `main` y la función `llamarFuncion`. La función `main` manda llamar a la función `llamarFuncion` dentro de un ciclo `for`. La función `llamarFuncion` crea una variable estática e imprime su valor.



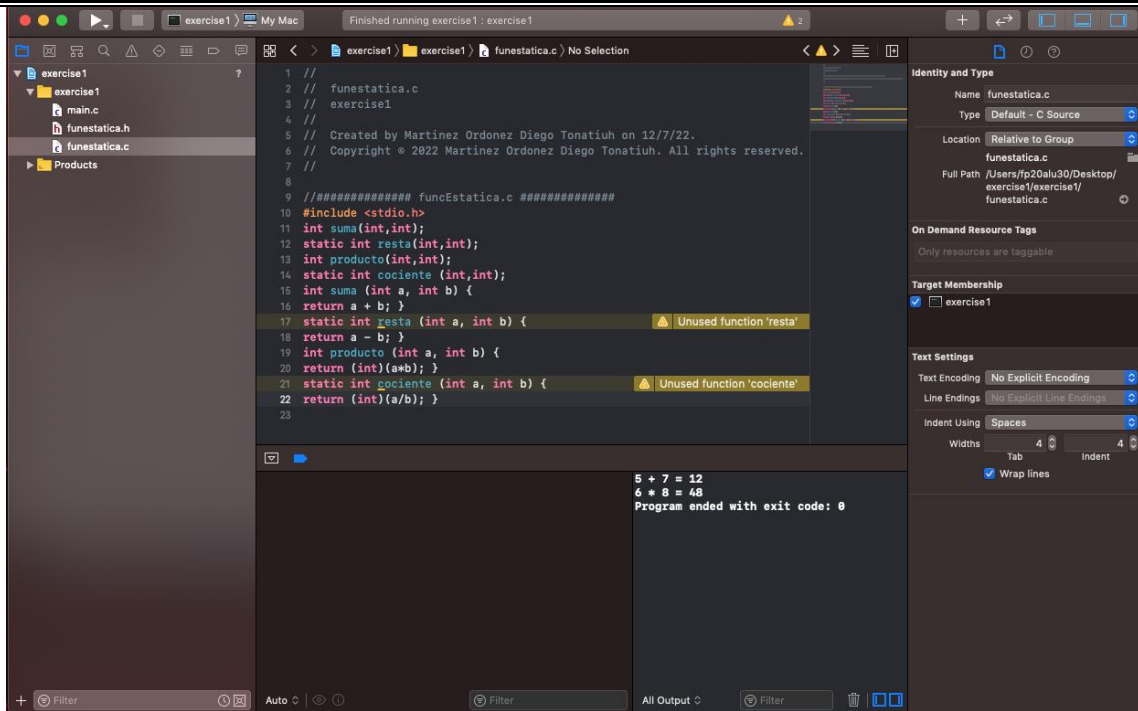
The screenshot shows a code editor with a C program. The code defines a static variable `numVeces` in the `llamarFuncion` function and calls it from `main` in a loop. The output shows the function being called 5 times.

```
1 //  
2 // main.c  
3 // exercise1  
4 //  
5 // Created by Martinez Ordenez Diego Tonatiuh on 12/7/22.  
6 // Copyright © 2022 Martinez Ordenez Diego Tonatiuh. All rights reserved.  
7 //  
8 //Te suma tu variable local con la global y de esta manera te da un valor  
9 //numérico//  
9 #include <stdio.h>  
10 void llamarFuncion();  
11 int main ()  
12 {  
13     for (int j=0 ; j < 5 ; j++)  
14     {  
15         llamarFuncion(); }  
16     }  
17     void llamarFuncion()  
18     {  
19         /* Solo la primera vez que se llame a esta función se creará y se le  
20          * asignará el valor de 0 a la variable estática numVeces */  
21         static int numVeces = 0;  
22         printf("Esta función se ha llamado %d veces.\n",++numVeces);  
23     }  
24 }
```

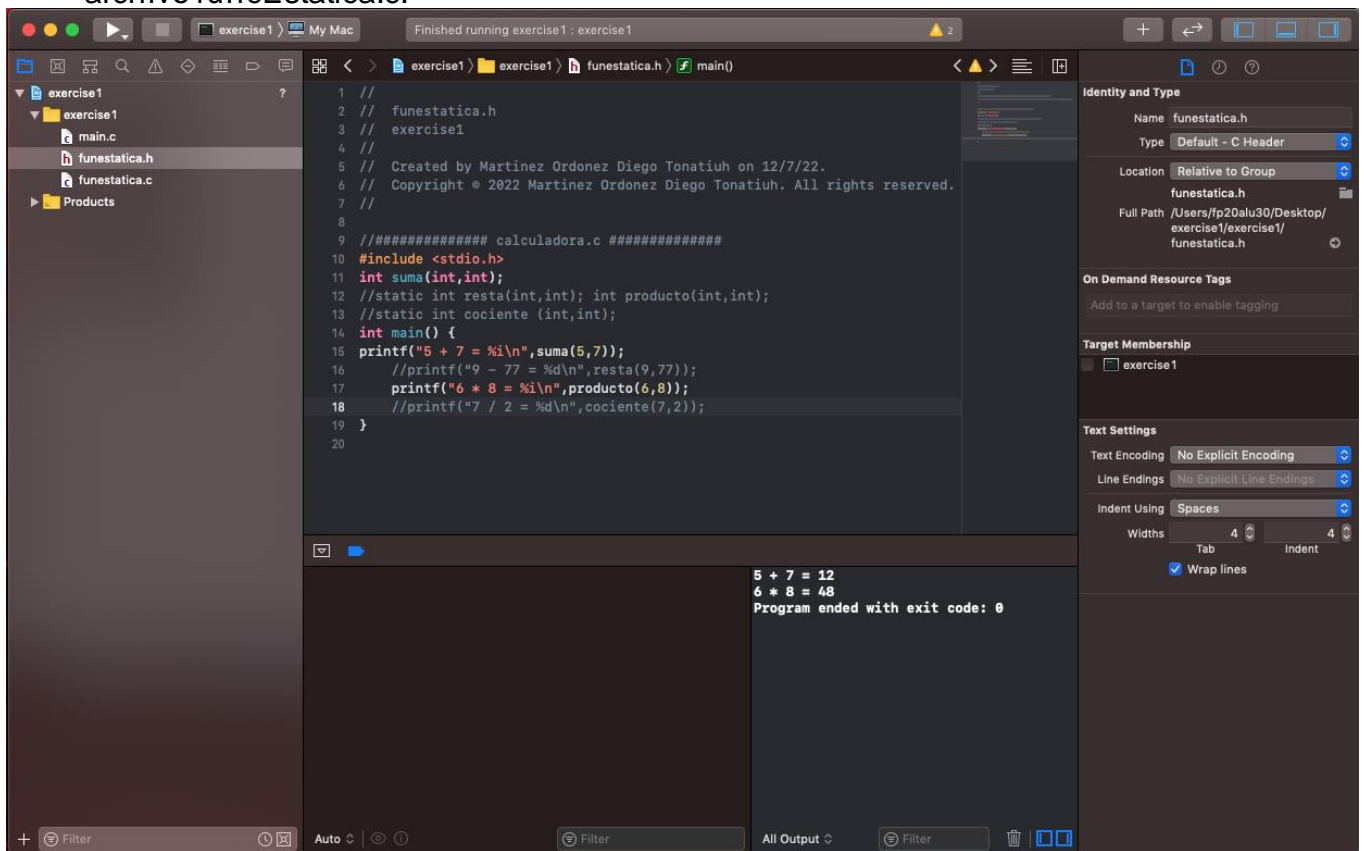
Output:

```
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.  
Program ended with exit code: 0
```

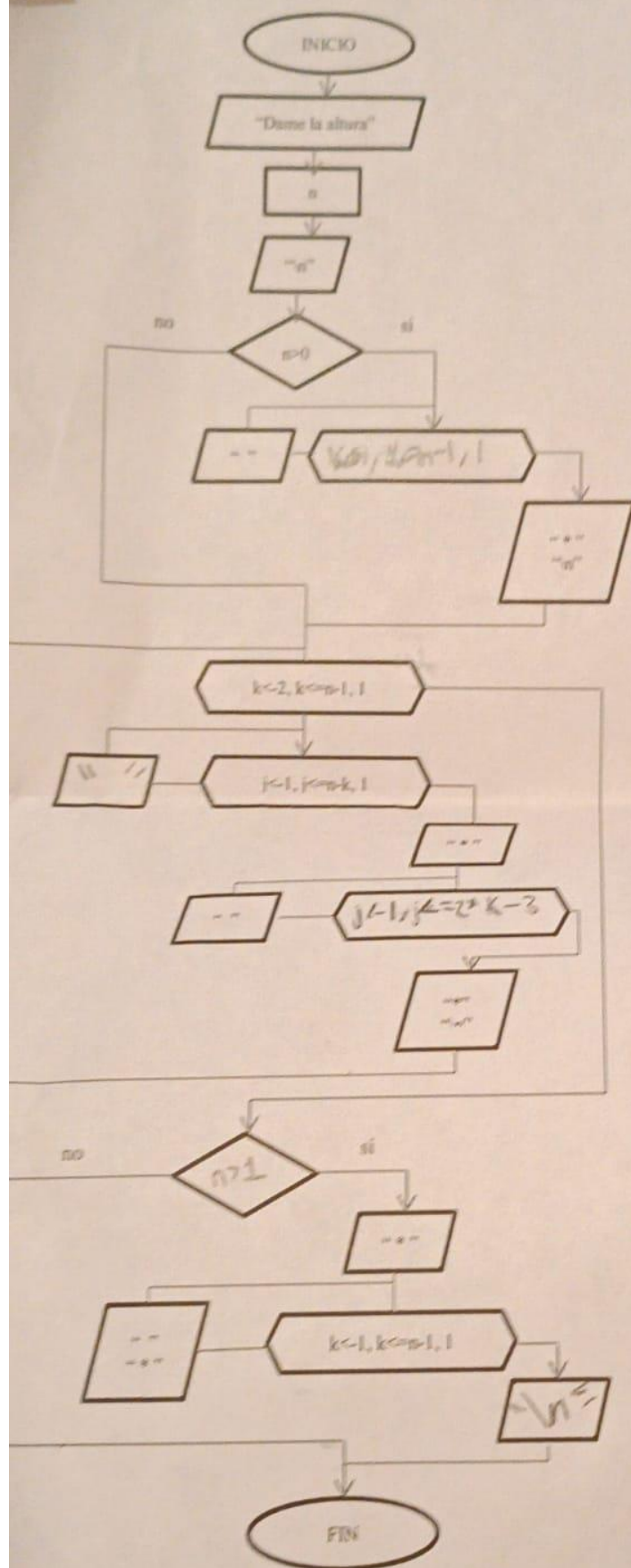
- El programa `funcEstatica.c` contiene las funciones de una calculadora básica: suma, resta, producto y cociente.



8. El programa calculadora.c contiene el método principal, el cual invoca a las funciones del archivo funcEstatica.c.



- Trabajo en clase:



Compara y completa las instrucciones en el diagrama de flujo y el pseudocódigo, escribe en la cuadrícula cual es la salida de los algoritmos para un valor de $n=10$ (Valor 5 puntos)

INICIO

ENTERO k, j, n ;

ESCRIBIR

LEER n ;

ESCRIBIR "\n";

SI $n > 0$ ENTONCES

PARA $k \leftarrow 1$ HASTA $k \leftarrow n-1$ CON PASO 1 HACER

ESCRIBIR " ";

FINPARA

ESCRIBIR

ESCRIBIR "\n";

FINSI

PARA $j \leftarrow 1$ HASTA $j \leftarrow n-k$ CON PASO 1 HACER

PARA $k \leftarrow 1$ HASTA $k \leftarrow j$ CON PASO 1 HACER

ESCRIBIR " ";

FINPARA

ESCRIBIR "++";

PARA $j \leftarrow 1$ HASTA $j \leftarrow 2*k-3$ CON PASO 1 HACER

ESCRIBIR " ";

FINPARA

ESCRIBIR "++";

ESCRIBIR "\n";

FINPARA

SI $n > 1$ ENTONCES

ESCRIBIR "++";

PARA $k \leftarrow 1$ HASTA

CON PASO 1 HACER

ESCRIBIR " ";

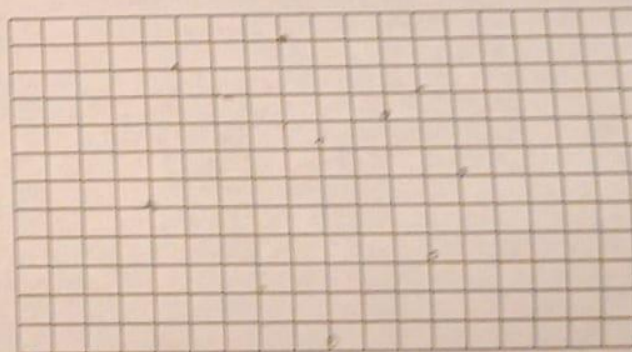
ESCRIBIR "++";

FINPARA

ESCRIBIR "\n";

FINSI

FIN



- **Conclusiones:**

Aprendimos nuevas cosas y a poner en práctica los conocimientos que nos ha brindado la materia a lo largo de este tiempo, también nos están dando apoyos y herramientas para poder trabajar de manera adecuada y que de esta manera sea mucho más sencillo el poder trabajar. En este caso estamos aprendiendo a usar adecuadamente las funciones y los tipos de estas que existen, ya que nos pueden brindar diferentes usos para los procesos y de esta forma el brindarnos lo mejor posible una base para poder hacer uso de ellas. Como podemos incluirlas en ciclos para que sean una herramienta que nos ayude a la realización de procesos y de cierta manera ya no tener que escribir todo de nuevo, así podemos ahorrarnos un tiempo y esfuerzo de cierta manera.

- **Referencias:**

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