



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* Marco Antonio Martínez Quintana

*Asignatura:* Fundamentos de Programación

*Grupo:* 1103

*No de Práctica(s):* 12

*Integrante(s):* Ulises Castro Rodríguez

*No. de Equipo de  
cómputo empleado:* No aplica

*No. de Lista o Brigada:* 08

*Semestre:* Primer Semestre

*Fecha de entrega:* Domingo 24/01/2021

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

# **Funciones.**

- **Objetivo**

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**

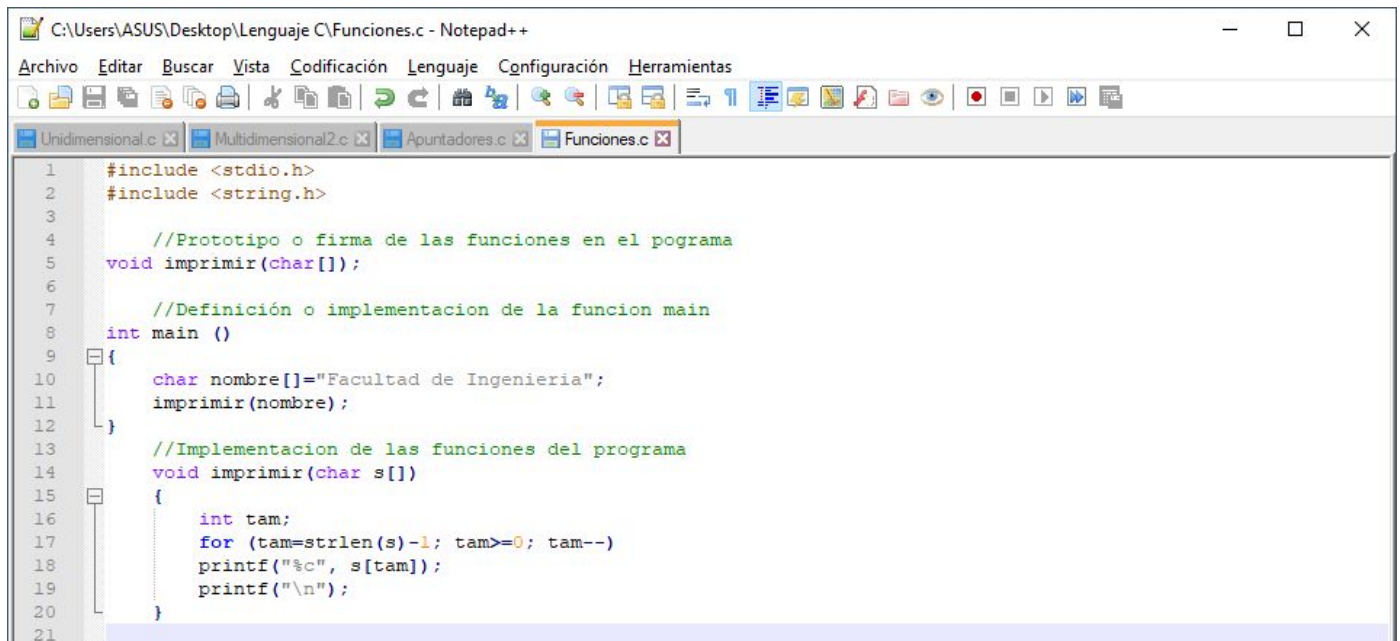
Como ya se mencionó, un programa en lenguaje C consiste en una o más funciones. C permite tener dentro de un archivo fuente varias funciones, esto con el fin de dividir las tareas y que sea más fácil la depuración, la mejora y el entendimiento del código.

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.

- **Actividades**

1. Elaborar en un programa en C la solución de un problema dividido en funciones.

## **Código (funciones)**

A screenshot of a Notepad++ window titled 'C:\Users\ASUS\Desktop\Lenguaje C\Funciones.c - Notepad++'. The window displays C code with line numbers 1 through 21. The code includes headers for stdio.h and string.h, defines a function prototype 'void imprimir(char[])', and implements the 'main' function and the 'imprimir' function. The 'main' function calls 'imprimir' with the string 'Facultad de Ingenieria'. The 'imprimir' function iterates through the string and prints each character on a new line.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  //Prototipo o firma de las funciones en el programa
5  void imprimir(char[]);
6
7  //Definición o implementación de la función main
8  int main ()
9  {
10     char nombre[]="Facultad de Ingenieria";
11     imprimir(nombre);
12 }
13 //Implementación de las funciones del programa
14 void imprimir(char s[])
15 {
16     int tam;
17     for (tam=strlen(s)-1; tam>=0; tam--)
18     printf("%c", s[tam]);
19     printf("\n");
20 }
21
```

```
Símbolo del sistema

C:\Users\ASUS\Desktop\Lenguaje C>gcc Funciones.c -o Funciones.exe

C:\Users\ASUS\Desktop\Lenguaje C>Funciones.exe
aireinegnI ed datlucaF

C:\Users\ASUS\Desktop\Lenguaje C>
```

## Código (ámbito de las variables)

```
1  #include <stdio.h>
2
3  void incremento();
4
5  int enteraGlobal=0;
6  int main()
7  {
8      for (int cont=0; cont<5; cont++)
9      {
10         incremento();
11     }
12     return 999;
13 }
14 void incremento()
15 {
16     int enteraLocal=5;
17     enteraGlobal+=2;
18     printf("global(%i)+local(%i)=%d\n", enteraGlobal, enteraLocal, enteraGlobal+enteraLocal);
19 }
```

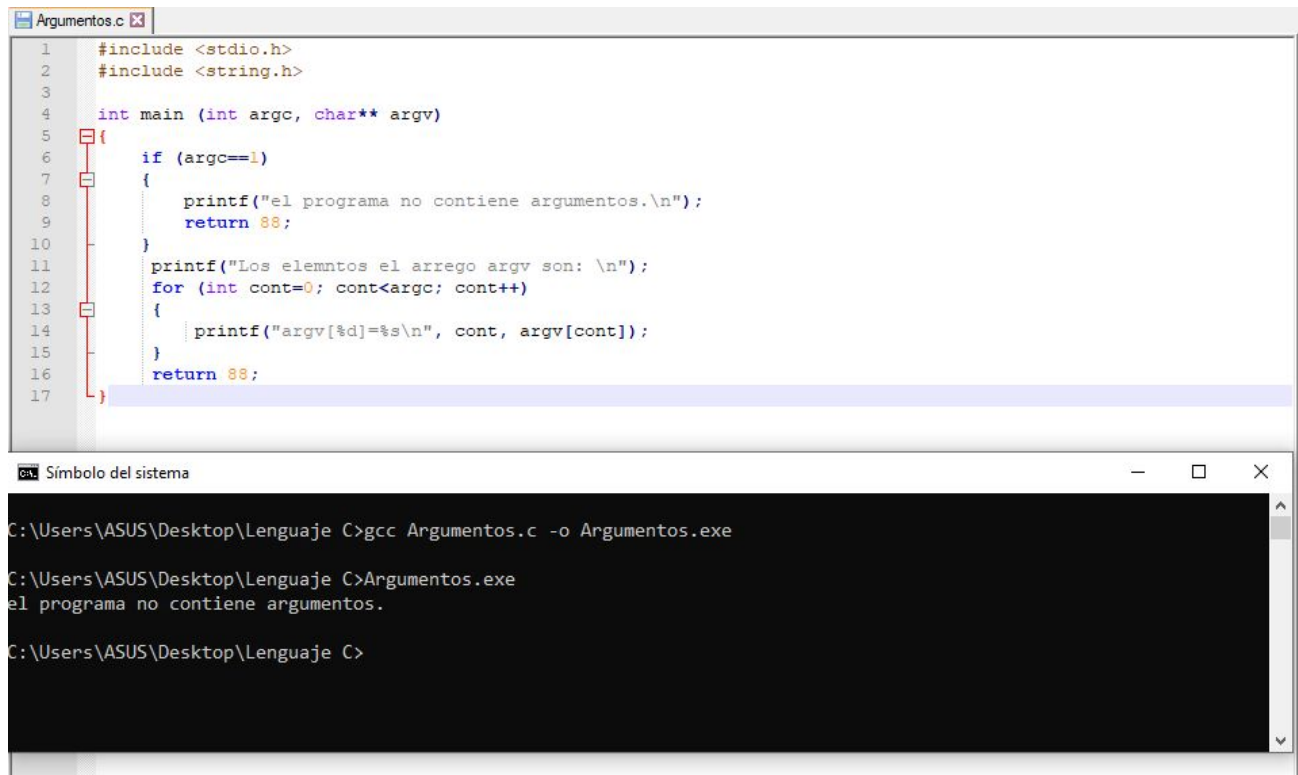
```
Símbolo del sistema

C:\Users\ASUS\Desktop\Lenguaje C>gcc Variables.c -o Variables.exe

C:\Users\ASUS\Desktop\Lenguaje C>Variables.exe
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
```

2. Elaborar un programa en C que maneje argumentos en la función principal.

### Código (argumentos funcion main)



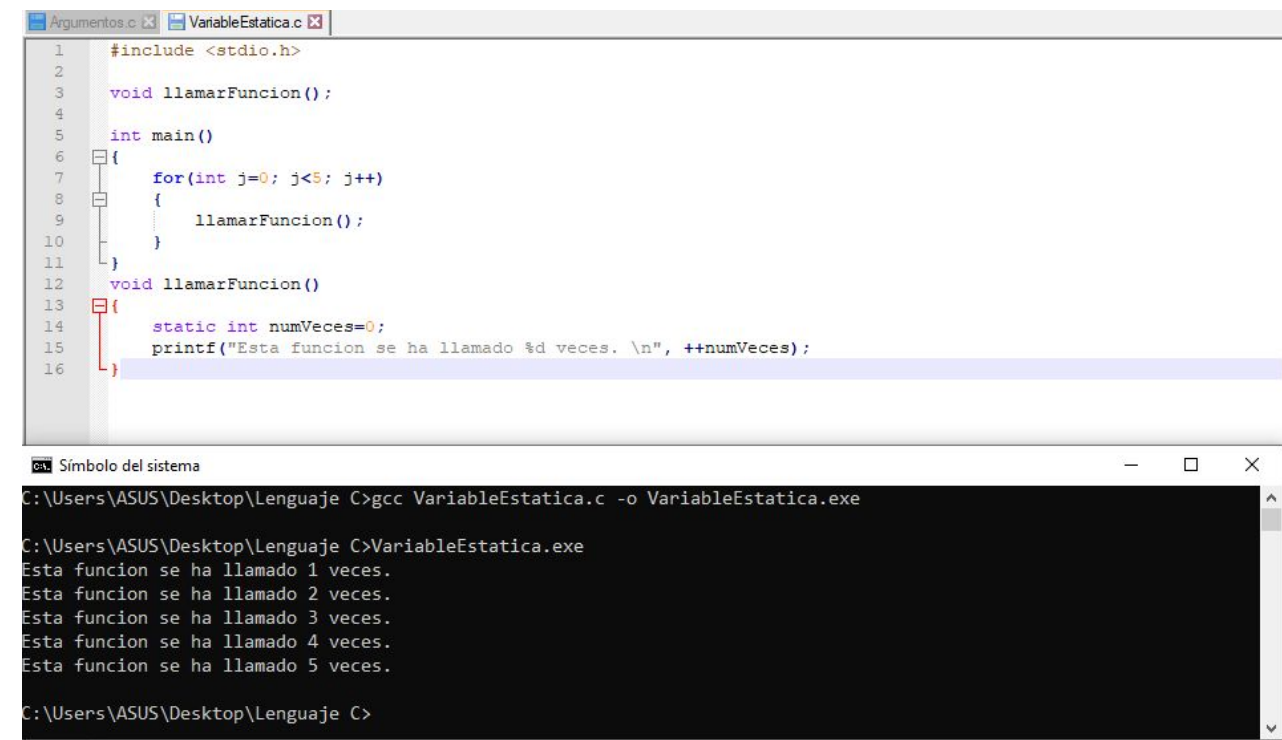
The screenshot shows a C program named 'Argumentos.c' in a text editor. The code includes `<stdio.h>` and `<string.h>`. The `main` function takes `argc` and `argv` as arguments. It checks if `argc` is 1; if so, it prints 'el programa no contiene argumentos.' and returns 88. Otherwise, it prints 'Los elemntos el arreglo argv son: \n', then iterates through `argv` from index 0 to `argc-1`, printing each element. Finally, it returns 88. Below the code, a terminal window shows the compilation command `gcc Argumentos.c -o Argumentos.exe` and the execution command `Argumentos.exe`, which results in the output 'el programa no contiene argumentos.'

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int main (int argc, char** argv)
5 {
6     if (argc==1)
7     {
8         printf("el programa no contiene argumentos.\n");
9         return 88;
10    }
11    printf("Los elemntos el arreglo argv son: \n");
12    for (int cont=0; cont<argc; cont++)
13    {
14        printf("argv[%d]=%s\n", cont, argv[cont]);
15    }
16    return 88;
17 }
```

```
C:\Users\ASUS\Desktop\Lenguaje C>gcc Argumentos.c -o Argumentos.exe
C:\Users\ASUS\Desktop\Lenguaje C>Argumentos.exe
el programa no contiene argumentos.
C:\Users\ASUS\Desktop\Lenguaje C>
```

3. En un programa en C, manejar variables y funciones estáticas.

### Código (Variable estática)

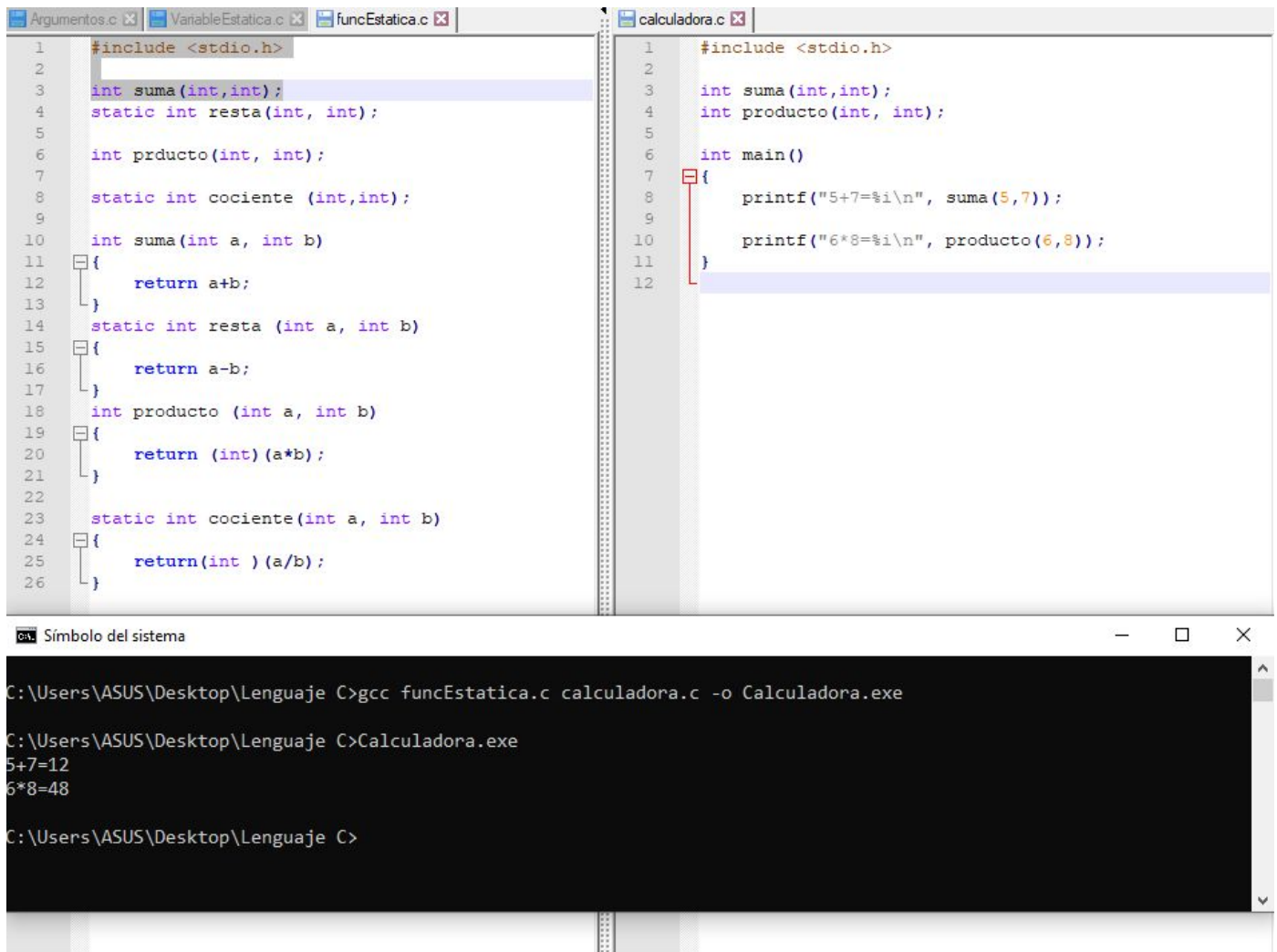


The screenshot shows a C program named 'VariableEstatica.c' in a text editor. It includes `<stdio.h>`. It defines a function `llamarFuncion()` and a `main` function. The `main` function has a loop that calls `llamarFuncion()` five times. The `llamarFuncion` function is defined with a static variable `numVeces` initialized to 0. Each time the function is called, it increments `numVeces` and prints 'Esta funcion se ha llamado %d veces.' followed by a newline. Below the code, a terminal window shows the compilation command `gcc VariableEstatica.c -o VariableEstatica.exe` and the execution command `VariableEstatica.exe`, which results in five lines of output, each showing the count increasing from 1 to 5.

```
1 #include <stdio.h>
2
3 void llamarFuncion();
4
5 int main()
6 {
7     for(int j=0; j<5; j++)
8     {
9         llamarFuncion();
10    }
11 }
12 void llamarFuncion()
13 {
14     static int numVeces=0;
15     printf("Esta funcion se ha llamado %d veces. \n", ++numVeces);
16 }
```

```
C:\Users\ASUS\Desktop\Lenguaje C>gcc VariableEstatica.c -o VariableEstatica.exe
C:\Users\ASUS\Desktop\Lenguaje C>VariableEstatica.exe
Esta funcion se ha llamado 1 veces.
Esta funcion se ha llamado 2 veces.
Esta funcion se ha llamado 3 veces.
Esta funcion se ha llamado 4 veces.
Esta funcion se ha llamado 5 veces.
C:\Users\ASUS\Desktop\Lenguaje C>
```

## Código (función estática)



The screenshot displays a C programming environment with two source files and a terminal window.

**funcEstatica.c**

```
1 #include <stdio.h>
2
3 int suma(int,int);
4 static int resta(int, int);
5
6 int producto(int, int);
7
8 static int cociente (int,int);
9
10 int suma(int a, int b)
11 {
12     return a+b;
13 }
14 static int resta (int a, int b)
15 {
16     return a-b;
17 }
18 int producto (int a, int b)
19 {
20     return (int) (a*b);
21 }
22
23 static int cociente(int a, int b)
24 {
25     return(int ) (a/b);
26 }
```

**calculadora.c**

```
1 #include <stdio.h>
2
3 int suma(int,int);
4 int producto(int, int);
5
6 int main()
7 {
8     printf("5+7=%i\n", suma(5,7));
9
10    printf("6*8=%i\n", producto(6,8));
11 }
12
```

**Símbolo del sistema**

```
C:\Users\ASUS\Desktop\Lenguaje C>gcc funcEstatica.c calculadora.c -o Calculadora.exe
C:\Users\ASUS\Desktop\Lenguaje C>Calculadora.exe
5+7=12
6*8=48
C:\Users\ASUS\Desktop\Lenguaje C>
```

### Conclusiones

El día de hoy, en esta práctica aprendimos acerca de las funciones en la programación en C, su función, su forma de utilizarlo etc, básicamente las funciones son una serie de comandos o serie de pasos que se almacenan en cierta parte del código. Otra cosa que aprendí al realizar la práctica fue la diferencia de las variables locales y globales, que básicamente la diferencia radica en que la variable está definida al inicio del código y la variable puede usarse en cualquier sitio de este; o puede estar definida en una función y esto provoca que la variable solo puede usarse en dicha función.

Por último esta práctica fue entretenida de hacer y abarca una parte importante de las bases de la programación, el uso de funciones para dividir el trabajo en secciones.

### • Bibliografías.

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