



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

<i>Profesor:</i>	M.I. Marco Antonio Martínez Quintana
<i>Asignatura:</i>	Fundamentos de Programación
<i>Grupo:</i>	03
<i>No de Práctica(s):</i>	12
<i>Integrante(s):</i>	Teran García Rodolfo Mario
<i>No. de Equipo de cómputo empleado:</i>	No aplica
<i>No. de Lista o Brigada:</i>	
<i>Semestre:</i>	2021-1
<i>Fecha de entrega:</i>	11/01/2021
<i>Observaciones:</i>	M.I. Marco Antonio Martínez Quintana

Calificación: \_\_\_\_\_

## Practica #12: Funciones

### 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:

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.

### Desarrollo:

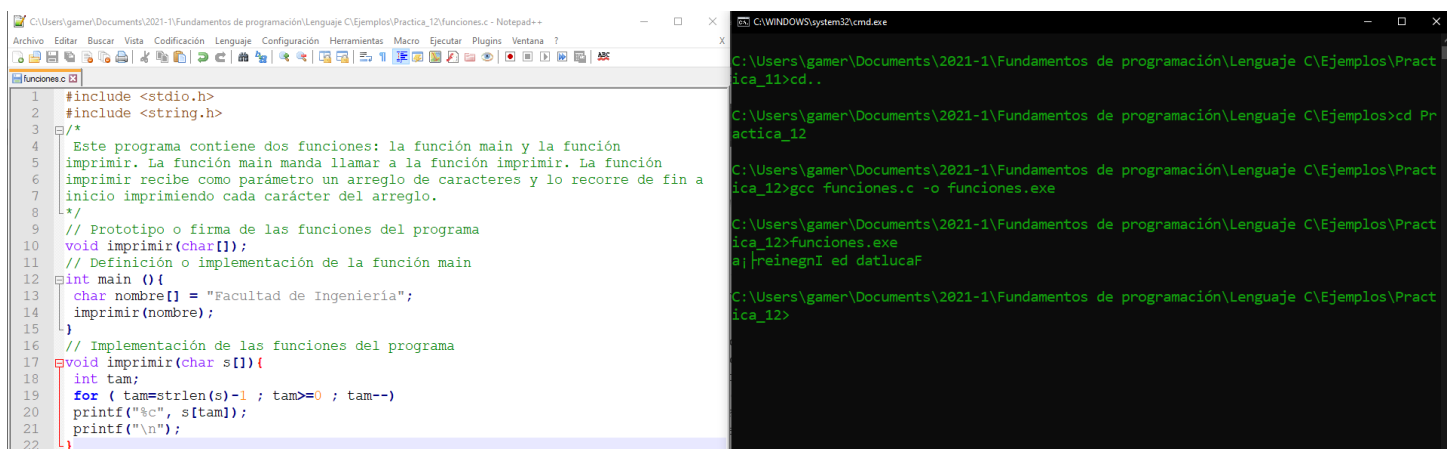
Una función puede recibir parámetros de entrada, los cuales son datos de entrada con los que trabajará la función, dichos parámetros se deben definir dentro de los paréntesis de la función, separados por comas e indicando su tipo de dato, de la siguiente forma:

(tipoDato nom1, tipoDato nom2, tipoDato nom3...)

El tipo de dato puede ser cualquiera de los vistos hasta el momento (entero, real, carácter o arreglo) y el nombre debe seguir la notación de camello. Los parámetros de una función son opcionales.

El valor de retorno de una función indica el tipo de dato que va a regresar la función al terminar el bloque de código de la misma. El valor de retorno puede ser cualquiera de los tipos de datos vistos hasta el momento (entero, real, carácter o arreglo), aunque también se puede regresar el elemento vacío (void). 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);

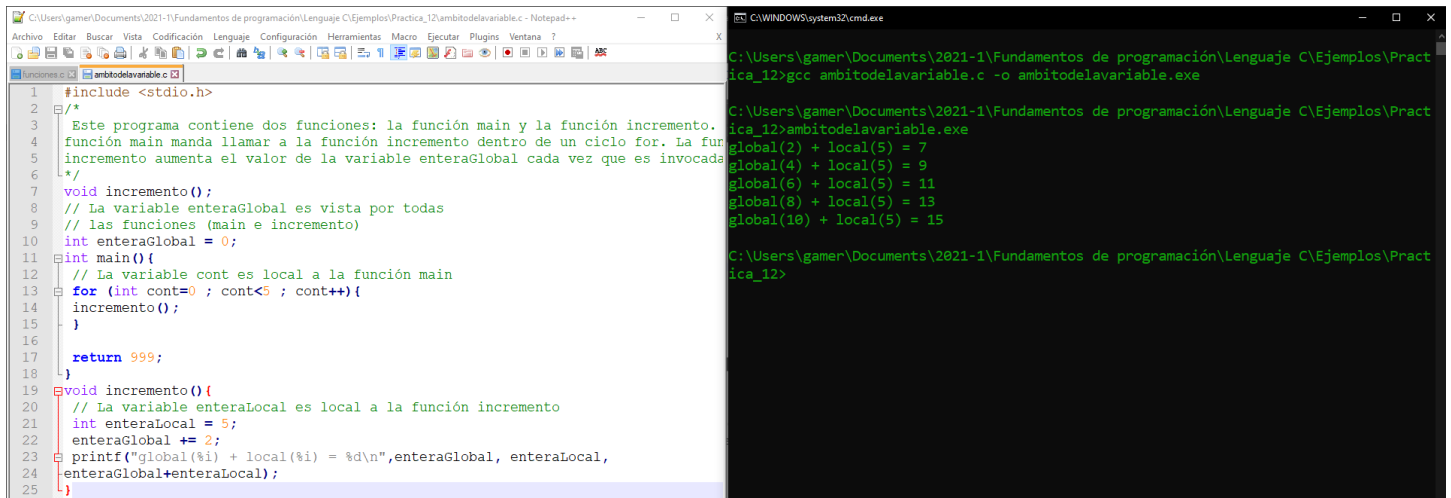


The image shows two side-by-side windows. The left window is a Notepad++ editor showing a C program named 'funciones.c'. The code includes `<stdio.h>` and `<string.h>`, defines a `main` function that calls `imprimir`, and implements `imprimir` to print a string in reverse. The right window is a Windows Command Prompt showing the execution of the program. It navigates to the directory containing the source file, compiles it with `gcc` to create `funciones.exe`, and then runs the executable, which outputs the string 'Facultad de Ingenieria' in reverse.

```
1 #include <stdio.h>
2 #include <string.h>
3
4 /*
5  Este programa contiene dos funciones: la función main y la función
6  imprimir. La función main manda llamar a la función imprimir. La función
7  imprimir recibe como parámetro un arreglo de caracteres y lo recorre de fin a
8  inicio imprimiendo cada carácter del arreglo.
9  */
10 // Prototipo o firma de las funciones del programa
11 void imprimir(char[]);
12 // Definición o implementación de la función main
13 int main () {
14     char nombre[] = "Facultad de Ingenieria";
15     imprimir(nombre);
16 }
17 // Implementación de las funciones del programa
18 void imprimir(char s[]) {
19     int tam;
20     for ( tam=strlen(s)-1 ; tam>=0 ; tam--)
21         printf("%c", s[tam]);
22     printf("\n");
23 }
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>cd ..
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos>cd Practica_12
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>gcc funciones.c -o funciones.exe
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>funciones.exe
a|reinegnI ed datlucaF
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>
```

**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.



```
#include <stdio.h>

/*
Este programa contiene dos funciones: la función main y la función incremento.
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.
*/

void incremento();
// La variable enteraGlobal es vista por todas
// las funciones (main e incremento)
int enteraGlobal = 0;

int main() {
    // La variable cont es local a la función main
    for (int cont=0 ; cont<5 ; cont++){
        incremento();
    }

    return 999;
}

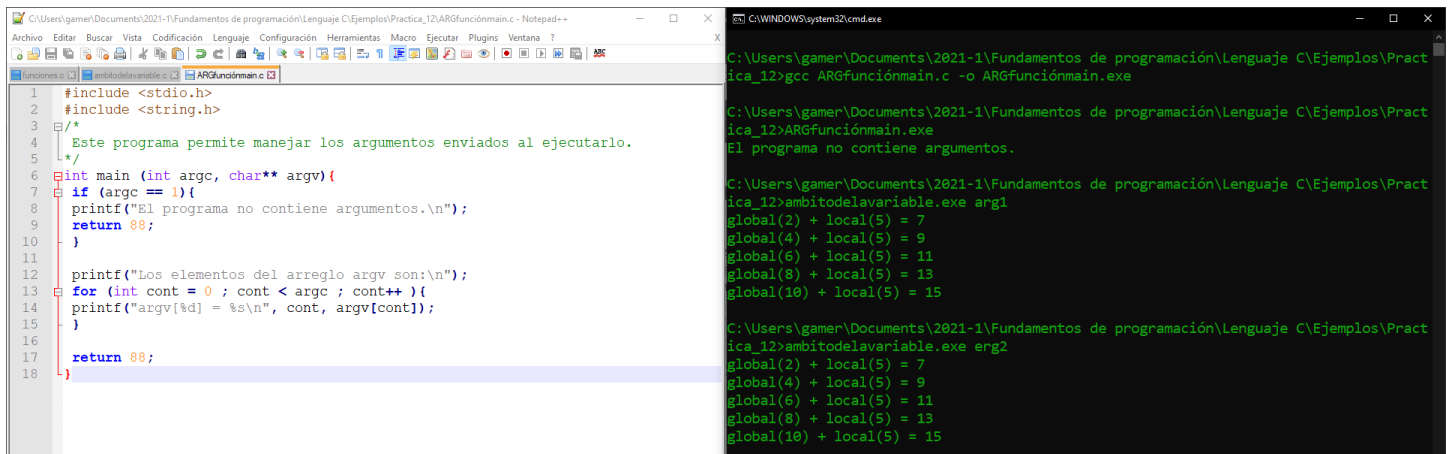
void incremento() {
    // La variable enteraLocal es local a la función incremento
    int enteraLocal = 5;
    enteraGlobal += 2;
    printf("global(%i) + local(%i) = %d\n", enteraGlobal, enteraLocal,
    enteraGlobal+enteraLocal);
}
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>gcc ambitoelavariab.c -o ambitoelavariab.exe

C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>ambitoelavariab.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

C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>
```

**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. 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).



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

/*
Este programa permite manejar los argumentos enviados al ejecutarlo.
*/

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;
}
```

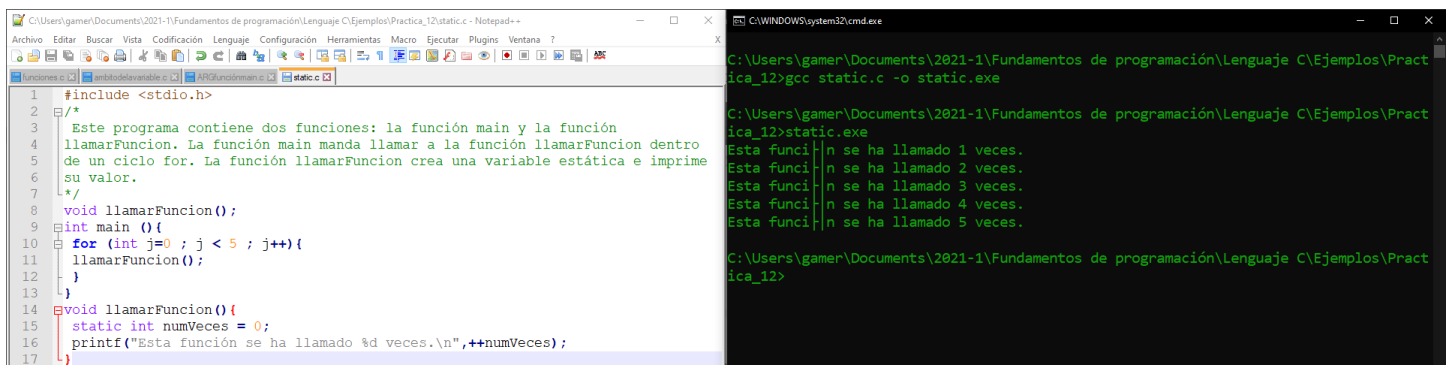
```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>gcc ARGfuncionmain.c -o ARGfuncionmain.exe

C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>ARGfuncionmain.exe
El programa no contiene argumentos.

C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>ambitoelavariab.exe arg1
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\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>ambitoelavariab.exe arg2
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
```

**Estático:** 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.



```
#include <stdio.h>

/*
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.
*/

void llamarFuncion();

int main() {
    for (int j=0 ; j < 5 ; j++){
        llamarFuncion();
    }
}

void llamarFuncion() {
    static int numVeces = 0;
    printf("Esta función se ha llamado %d veces.\n", ++numVeces);
}
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>gcc static.c -o static.exe

C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>static.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\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>
```

```
//##### funcEstatica.c #####
#include <stdio.h>
/*
Este programa contiene las funciones de una calculadora básica: suma, resta, producto, cociente.
*/
int suma(int,int);
static int resta(int,int);
int producto(int,int);
static int cociente(int,int);
int suma (int a, int b){
    return a + b;
}
static int resta (int a, int b){
    return a - b;
}
int producto (int a, int b){
    return (int)(a*b);
}
static int cociente (int a, int b){
    return (int)(a/b);
}

//##### calculadora.c #####
#include <stdio.h>
/*
Este programa contiene el método principal, el cual invoca a las funciones
del archivo funcEstatica.c.
*/
int suma(int,int);
//static int resta(int,int);
int producto(int,int);
//static int cociente(int,int);
int main(){
    printf("5 + 7 = %i\n",suma(5,7));
    //printf("9 - 77 = %d\n",resta(9,77));
    printf("6 * 8 = %i\n",producto(6,8));
    //printf("7 / 2 = %d\n",cociente(7,2));
}
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>gcc funcEstatica.c calculadora.c -o exe
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>gcc funcEstatica.c calculadora.c -oexe
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>exe.exe
5 + 7 = 12
6 * 8 = 48
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>exe.exe
5 + 7 = 12
6 * 8 = 48
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>
```

```
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>gcc funcEstatica.c calculadora.c -o exe
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>gcc funcEstatica.c calculadora.c -oexe
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>exe.exe
5 + 7 = 12
6 * 8 = 48
C:\Users\gamer\Documents\2021-1\Fundamentos de programación\Lenguaje C\Ejemplos\Practica_12>
```

Conclusiones:

Como en las prácticas anteriores esta me ha dejado un conocimiento invaluable, del cual siempre logro sorprenderme por lo mucho que, pese a tener una noción del funcionamiento de una computadora, aún así desconozco, sin lugar a dudas, la computadora es el invento del siglo, pues reúne las aplicaciones de conceptos ingenieriles y matemáticos en una abstracción tal que podemos resumir todo en unos y ceros, pero visto desde consola y tras haber estudiado todos y cada uno de los programas propuestos hasta la fecha, determino de manera casi inmediata que esta práctica en particular alberga cierta abstractividad muy susceptible a la interpretación humana, pues nos abre las puertas hacia un terreno muy poco explorado (al menos por mí), debido a que normalmente en las estructuras de cualquier programa solemos insertar la función main seguida de dos paréntesis de manera casi automática sin analizar a fondo lo que realmente significa, hasta ahora es que me doy cuenta de lo mucho que importa cada segmento en la realización de un código, pues cada uno importa y es sumamente necesario, por ello también es necesario continuar por el camino del aprendizaje y darle más esmero a todas y cada una de las prácticas, hasta llegar a ese punto en el que cada vez sean menos las cosas que nos sorprendan, pues conoceremos cada vez más profundo el mundo de la programación. Y pensar que este es solo el lenguaje C que, aunque muchos comparten cosas similares tal vez en su sintaxis o estructuras de ejecución, realmente es increíble pensar que estamos tratando de conquistar una isla de entre muchas que existen en este archipiélago computacional de lenguajes destinados a la codificación y ejecución de programas.