



## Sistema de gestión de calidad

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* Marco Antonio Martínez Quintana

*Asignatura:* Fundamentos de programación

*Grupo:* 3

*No. de Práctica(s):* #12

*Integrante(s):* María Guadalupe Martínez Pavón

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

*No. de Lista o*

*Semestre:* 1

*Fecha de entrega:* 29-01-2021

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

## Guía práctica de estudio 12: 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.

**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.

### Introducción:

#### Como lo veíamos anteriormente siempre a la hora de crear

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, 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.

### Código (funciones)

```
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  
11 // Prototipo o firma de las funciones del programa  
12 void imprimir(char[]);  
13 // Definición o implementación de la función main  
14 int main () {  
15     char nombre[] = "Facultad de Ingenieria";  
16     imprimir(nombre);  
17 }  
18 // Implementación de las funciones del programa  
19 void imprimir(char s[]){  
20     int tam;  
21     for ( tam=strlen(s)-1 ; tam>=0 ; tam-- )  
22         printf("%c", s[tam]);  
23     printf("\n");  
24 }
```

```
Símbolo del sistema
C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>gcc Fun.c -o Fun.exe
C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>Fun.exe
aj|reinegni ed datlucaF
C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>
```

## Código (Ámbito de las variables)

```
1 #include <stdio.h>
2 /*
3  Este programa contiene dos funciones: la función main y la función incremento. La
4  función main manda llamar a la función incremento dentro de un ciclo for. La función
5  incremento aumenta el valor de la variable enteraGlobal cada vez que es invocada.
6  */
7
8 void incremento();
9
10 // La variable enteraGlobal es vista por todas
11 // las funciones (main e incremento)
12 int enteraGlobal = 0;
13 int main(){
14     // La variable cont es local a la función main
15     for (int cont=0 ; cont<5 ; cont++){
16         incremento();
17     }
18
19     return 999;
20 }
21 void incremento() {
22
23     // La variable enteraLocal es local a la función incremento
24     int enteraLocal = 5;
25     enteraGlobal += 2;
26     printf("global(%i) + local(%i) = %d\n",enteraGlobal, enteraLocal,
27     enteraGlobal+enteraLocal);
28 }
```

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.18363.1316]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\lupit>cd OneDrive
C:\Users\lupit\OneDrive>cd Escritorio
C:\Users\lupit\OneDrive\Escritorio>cd "Lenguaje C"
C:\Users\lupit\OneDrive\Escritorio\Lenguaje C>cd Ejemplos
C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>gcc ambito.c -o ambito.exe
C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>ambito.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\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>
```

## Código (argumentos función main)

```
1 #include <stdio.h>
2 #include <string.h>
3 /*
4 | Este programa permite manejar los argumentos enviados al ejecutarlo.
5 */
6 int main (int argc, char** argv){
7     if (argc == 1){
8         printf("El programa no contiene argumentos.\n");
9         return 88;
10    }
11
12    printf("Los elementos del arreglo argv son:\n");
13    for (int cont = 0 ; cont < argc ; cont++){
14        printf("argv[%d] = %s\n", cont, argv[cont]);
15    }
16
17    return 88;
18 }
```

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.18363.1316]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\lupit>cd OneDrive

C:\Users\lupit\OneDrive>cd Escritorio

C:\Users\lupit\OneDrive\Escritorio>cd "Lenguaje C"

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C>cd Ejemplos

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>gcc main.c -o main.exe

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>main.exe
El programa no contiene argumentos.

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>
```

## Código (variable estática)

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

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.18363.1316]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\lupit>cd OneDrive

C:\Users\lupit\OneDrive>cd Escritorio

C:\Users\lupit\OneDrive\Escritorio>cd "Lenguaje C"

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C>cd Ejemplos

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>gcc estatica.c -o estatica.exe

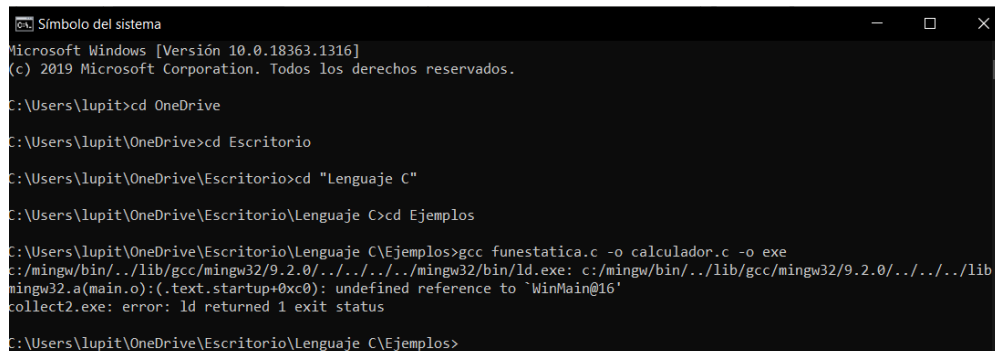
C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>estatica.exe
Esta funci n se ha llamado 1 veces.
Esta funci n se ha llamado 2 veces.
Esta funci n se ha llamado 3 veces.
Esta funci n se ha llamado 4 veces.
Esta funci n se ha llamado 5 veces.

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>
```

## Código (función estática)

```
1 //##### funcEstatica.c #####
2 #include <stdio.h>
3
4 /*
5 Este programa contiene las funciones de una calculadora básica: suma, resta, producto y
6 cociente.
7 */
8 int suma(int,int);
9 static int resta(int,int);
10
11 int producto(int,int);
12
13 static int cociente (int,int);
14
15
16
17 int suma (int a, int b){
18     return a + b;
19 }
20
21 static int resta (int a, int b){
22     return a - b;
23 }
24 int producto (int a, int b){
25     return (int)(a*b);
26 }
27 static int cociente (int a, int b){
28     return (int)(a/b);
29 }
```

```
1 //##### calculadora.c #####
2 #include <stdio.h>
3
4 /*
5 Este programa contiene el método principal, el cual invoca a las funciones
6 del archivo funcEstatica.c.
7 */
8 int suma(int,int);
9 //static int resta(int,int);
10 int producto(int,int);
11 //static int cociente (int,int);
12
13 int main(){
14     printf("5 + 7 = %i\n",suma(5,7));
15     //printf("9 - 77 = %d\n",resta(9,77));
16     printf("6 * 8 = %i\n",producto(6,8));
17     //printf("7 / 2 = %d\n",cociente(7,2));
18 }
```



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.18363.1316]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\lupit>cd OneDrive

C:\Users\lupit\OneDrive>cd Escritorio

C:\Users\lupit\OneDrive\Escritorio>cd "Lenguaje C"

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C>cd Ejemplos

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>gcc funestatica.c -o calculador.c -o exe
c:/mingw/bin/./lib/gcc/mingw32/9.2.0/./../././mingw32/bin/ld.exe: c:/mingw/bin/./lib/gcc/mingw32/9.2.0/./.././lib
mingw32.a(main.o):(.text.startup+0xc0): undefined reference to `WinMain@16'
collect2.exe: error: ld returned 1 exit status

C:\Users\lupit\OneDrive\Escritorio\Lenguaje C\Ejemplos>
```

## Conclusión

La utilización de funciones nos permite dividir un programa extenso en pequeños segmentos que realizan tareas concretas, con estas mismas funciones se puede que dentro de un mismo programa se realicen las mismas tareas varias veces, lo que se facilita mediante la utilización de funciones, tienen el propósito de permitir un manejo eficiente de los datos, las funciones en C no se pueden anidar. En otras palabras, una función no se puede declarar dentro de otra función, por lo que todas las funciones son globales o externas, lo que hace que puedan llamarse desde cualquier parte de un programa.

## Bibliografía

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