

	<b>Carátula para entrega de prácticas</b>	
Facultad de Ingeniería	Laboratorio de docencia	

# Laboratorios de computación salas A y B

*Profesor:* Karina García Morales

*Asignatura:* Fundamentos de programación

*Grupo:* 20

*No. de práctica(s):* 11

*Integrante(s):* Suzán Herrera Álvaro

*No. de lista o brigada:* 49

*Semestre:* 1

*Fecha de entrega:* 13 de diciembre 2022

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

**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.

Conceptos

Funciones: Elemento agrupador de un bloque de acciones que puede utilizarse para realizar operaciones y con ello ahorrar tiempo en código

- TipoDato fnNombre (entradas){• • •}

Código (funciones)

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.

```
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--)| ⚠ Implicit conversion loses integer precision: 'unsigned long' to 'int'
16     printf("%c", s[tam]);
17     printf("\n");
18 }
19
20 int main (){
21     char nombre[] = "Facultad de Ingeniería";
22     imprimir(nombre);
23 }
24 // Implementación de las funciones del programa
25
```



```
a\201\314ireinegnI ed datlucaF
Program ended with exit code: 0
```

Auto ↕ | 🔍 ⓘ

🔍 Filter

All Output ↕

🔍 Filter



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

```
1 // Prototipo o firma de las funciones del programa
2 void imprimir(char[]);
3 // Definición o implementación de la función main
4 int main(){
5     char nombre[] = "Facultad de Ingeniería"; imprimir(nombre);
6 }
7 void imprimir(char s[]){ int tam;
8     for (tam=0; tam<=strlen(s)-1 ; tam++)
9         printf("%c", s[tam]);
10    printf("\n");
11 }
```



```
Facultad de Ingeniería
Program ended with exit code: 0
```

Código (variables locales) El siguiente programa muestra la declaración y uso de variables locales en la función de suma.

```
8
9 #include <stdio.h>
10 int z; //Variable global
11 void sumar();
12 int main() {
13     sumar(); // llamado de la función suma
14     z=z+1;
15 }
16 void sumar() // función suma
17 {
18     int x=5, y=10; //variables locales
19     z=x+y;
20     printf("%i\n",z);
21 }
22
```

This function declaration is not a prototype

```
15
Program ended with exit code: 0
```

Código (variables globales) El siguiente programa muestra la declaración de la variable global resultado, la cual es utilizada en ambas funciones.

Practica 11 > Practica 11 > main.c > main()

```
1 #include <stdio.h>
2 int resultado; //variable global
3 int main() {
4     multiplicar(); //llamado de la función multiplicar
5     printf("%i\n\n", resultado);
6     return 0;
7 }
8 int multiplicar() //función multiplicar
9 {
10     resultado = 5 * 4;
11     return 0;
12 }
13
```

2 ⚠ Implicit declaration of function 'multiplicar' is invalid in C99

20

Program ended with exit code: 0

Código (Ámbito de las variables) 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.

```
8
9 #include <stdio.h>
10 #include<stdio.h>
11 void incremento(); ⚠ This function declaration is not a prototype
12 /* La variable enteraGlobal es vista por todas
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 }
25 return 999;
26
27 }
28 void incremento() {
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, enteraGlobal+enteraLocal);
33     //return 8; Esta línea de código no se debe colocar debido a que el tipo de dato de
```

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

Auto ↕ | 🔍 Filter | All Output ↕ | 🔍 Filter | 🗑️ | 📄 | 📄 | 📄

Código (argumentos función main) Este programa muestra los argumentos enviados al ejecutarlo.

```

1 #include <stdio.h>
2 #include <stdio.h>
3 int cont;
4 int incremento();
5 /* La variable enteraGlobal es vista por todas
6 las funciones (main e incremento) */
7 int enteraGlobal;
8 int main() {
9     // La variable cont es local a la función main
10    enteraGlobal = 0; // La función main accede a la variable global
11    for (cont=0 ; cont<5 ; cont++){
12        incremento(); }
13    return 999; }
14 int incremento() {
15     // La variable enteraLocal es local a la función incremento
16    int enteraLocal = 5;
17    enteraGlobal += 2;
18    printf("global(%i) + local(%i) = %d\n", enteraGlobal, enteraLocal, enteraGlobal+enteraLocal);
19    return 0; }
20 /* La variable global aumenta 2 y se le suma la variable local, muestra el resultado y luego incrementa en 2 la variable
21    global; el ciclo se repite 5 veces debido al for*/
21

```

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

Código (variable estática) 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.

```

Last login: Wed Dec 7 19:11:31 on console
[Etiopia41:~ fp20alu49$ vi programa4.c
[Etiopia41:~ fp20alu49$ gcc programa4.c -o programa4.out
[Etiopia41:~ fp20alu49$ vi programa4.c
[Etiopia41:~ fp20alu49$ ./programa4.out
El programa no contiene argumentos.
[Etiopia41:~ fp20alu49$ Álvaro Suzán
-bash: Álvaro: command not found
[Etiopia41:~ fp20alu49$ ./programa4.out
El programa no contiene argumentos.
[Etiopia41:~ fp20alu49$ a l v a r o s u z a n
-bash: a: command not found
[Etiopia41:~ fp20alu49$ ./programa4.out
El programa no contiene argumentos.
[Etiopia41:~ fp20alu49$ a b c
-bash: a: command not found
[Etiopia41:~ fp20alu49$ ./programa4.out
El programa no contiene argumentos.
[Etiopia41:~ fp20alu49$ ./programa4.out alvaro suzan
Los elementos del arreglo argv son:
argv[0] = ./programa4.out
argv[1] = alvaro
argv[2] = suzan
[Etiopia41:~ fp20alu49$

```

Código (función estática). Este ejemplo consta de dos archivos: funcEstatica.c y calculadora.c. El programa funcEstatica.c contiene las funciones de una calculadora básica: suma, resta, producto y cociente



```

7 //
8 #include <stdio.h>
9 void llamarFuncion();
10 int main () {
11     for (int j=0 ; j < 5 ; j++) {
12         llamarFuncion(); }
13     }
14     void llamarFuncion() {
15         /* Solo la primera vez que se llame a esta función se creará y se le asignará el valor de 0
16            a la variable estática numVeces */
17         static int numVeces = 0;
18         printf("Esta función se ha llamado %d veces.\n",++numVeces);
19     }

```

⚠ This function declaration is not a prototype

```

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

```

The screenshot shows a code editor with the following code in 'funcEstatica.c':

```

1 //##### funcEstatica.c #####
2 #include <stdio.h>
3 int suma(int,int);
4 //static int resta(int,int);
5 int producto(int,int);
6 int cociente (int,int);
7 int suma (int a, int b) {
8     return a + b; }
9
10 int producto (int a, int b) {
11     return (int)(a*b); }
12 int cociente (int a, int b) {
13     return (int)(a/b); }
14

```

Below the code editor, the output of the program is displayed:

```

5 + 7 = 12
9 - 77 = -68
6 * 8 = 48
7 / 2 = 3
Program ended with exit code: 0

```

```
// Created by Perez Alatorre Erick Roberto on 12/7/22.
// Copyright © 2022 Perez Alatorre Erick Roberto. All rights reserved.
// FUNCION DEL PROGRAMA
// ===== calculadora.c =====
#include <stdio.h>
int suma(int, int);
static int resta(int, int);
int producto(int, int);
static int resta (int a, int b)
{
    return a - b;
}
int cociente (int, int);
int main() {
    printf("8 + 7 = %d\n", suma(8, 7));
    printf("9 - 77 = %d\n", resta(9, 77));
    printf("6 * 8 = %d\n", producto(6, 8));
    printf("7 / 2 = %d\n", cociente(7, 2));
}
```

8 + 7 = 15  
9 - 77 = -68  
6 \* 8 = 48  
7 / 2 = 3.5  
Program ended with exit code: 0

## Tarea

Pruebas de escritorio; programa que muestra un triángulo en ASCII recopilando solamente la altura, el diagrama y el pseudocódigo se completan y se analizan; el programa fue modificado para trabajar con valores positivos en los ciclos for.

```
// Libraries
#include <stdio.h>
#include <stdlib.h>
// Global variables
int n, j, k;
// MAIN
int main() {
    printf("Dame la altura: ");
    scanf("%d", &n);
    printf("\n");
    if(n > 0) {
        for(k = 0; k <= n; k++) {
            printf(" ");
        }
        printf("**");
        printf("\n");
    }
    for(k = 0; k <= n; k++) {
        for(j = 0; j <= n - k; j++) {
            printf(" ");
        }
        printf("**");
        for(j = 0; j <= (2 * k); j++) {
            printf(" ");
        }
        printf("**");
        printf("\n");
    }
    if(n > 1) {
        printf("**");
        for(k = 0; k <= n; k++) {
            printf(" ");
        }
        printf("**");
        printf("\n");
    }
    return 0;
}
```

> make -s  
> ./main  
Dame la altura: 10

```

      *
     **
    ***
   ****
  *****
 *****
  *****
   ****
    ***
     **
      *
```

Diagrama de flujo y pseudocódigo

Alvaro Suzain

```

graph TD
    INICIO([INICIO]) --> Input[/Dame la altura/]
    Input --> n[n]
    n --> n0{n > 0}
    n0 -- no --> FIN([FIN])
    n0 -- si --> k1[k <- 1, k <= n-1, 1]
    k1 --> j1[j <- 1, j <= n-k, 1]
    j1 --> jgt1{j > 1}
    jgt1 -- no --> FIN
    jgt1 -- si --> k2[k <- 1, k <= n-1, 1]
    k2 --> j2[j <- 1, j <= n-k-1, 1]
    j2 --> jgt1
    jgt1 -- no --> FIN
  
```

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 "Dame la altura";

LEER  $n$ ;

ESCRIBIR "\n";

SI  $n > 0$  ENTONCES

PARA  $k <- 1$  HASTA  $k <= n-1$  CON PASO 1 HACER

ESCRIBIR " ";

FINPARA

ESCRIBIR " + ";

ESCRIBIR "\n";

FINSI

PARA  $k <- 2$  HASTA  $k <= n-1$  CON PASO 1 HACER

PARA  $j <- 1$  HASTA  $j <= n-k$

CON PASO 1 HACER

ESCRIBIR " ";

FINPARA

ESCRIBIR " + ";

PARA  $j <- 1$  HASTA  $j <= 2 * k - 3$  CON PASO 1 HACER

ESCRIBIR " ";

FINPARA

ESCRIBIR " + ";

ESCRIBIR "\n";

FINPARA

SI  $n > 1$  ENTONCES

ESCRIBIR " + ";

PARA  $k <- 1$  HASTA  $k <= n-1$

CON PASO 1 HACER

ESCRIBIR " ";

ESCRIBIR " + ";

FINPARA

ESCRIBIR "\n";

FINSI

FIN

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  $k < -1$  HASTA  $k \leq n-1$  CON PASO 1 HACER

ESCRIBIR " ";

**FINPARA**

ESCRIBIR 

```
ESCRIBIR "\n";
```

**FINSI**

PARA  $k \leftarrow 2$  HASTA  $k \leq n-1$  CON PASO 1 HACER

PARAJ&lt;-1HASTA

CON PASO 1 HA

CON PASO Y HACER  
ESCRIBIR

**FINPARA**

ESCRIBIR "•";

PARA  $j < -1$  HASTA  $j \leq 2 \cdot k - 3$  CON PASO 1 HACER

ESC

ESCRIBIR "•";

ESCRIBIR "\n";

**FINPARA**

SI  $n > 1$  ENTONCES

ESCRIBIR <sup>10</sup> 100;

PARAk&lt;-1HASTA

CON PASO 1 HACER

ESCRIBIR

ES

**FINPARA**

**FINSI**

FIN

"Dame la a l'ora"

## Conclusión:

Las Funciones tienen diversos usos en muchos aspectos de ingeniería, en este caso de programación sirven para recopilar los datos dentro de un programa que requiere varias veces la misma estructura y así ahorrar tiempo al momento de ejecutar los bloques y declararlos, se puede realizar por medio de varios archivos .C dentro de uno solo declarándolos antes de usarlos.

Esta práctica será muy útil en futuros programas que tengamos que hacer.

<https://github.com/3201050964/Pr-ctica-11>