



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* Karina Morales Garcia

*Asignatura:* Fundamentos de programacion

*Grupo:* 20

*No. de práctica(s):* 11

*Integrante(s):* Avila Pineda Samuel David

*No. de lista o brigada:* 06

*Semestre:* 2023-1

*Fecha de entrega:* 14 de diciembre del 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.

### 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.

### ¿Cuál es la función principal del lenguaje C?

Como ya lo habíamos visto es la función main. Cuando se ordena la ejecución del programa, se inicia con la ejecución de las instrucciones que están dentro de la función y a la misma vez puede llamar a ejecutar otras funciones.

### Funciones

La sintaxis es:

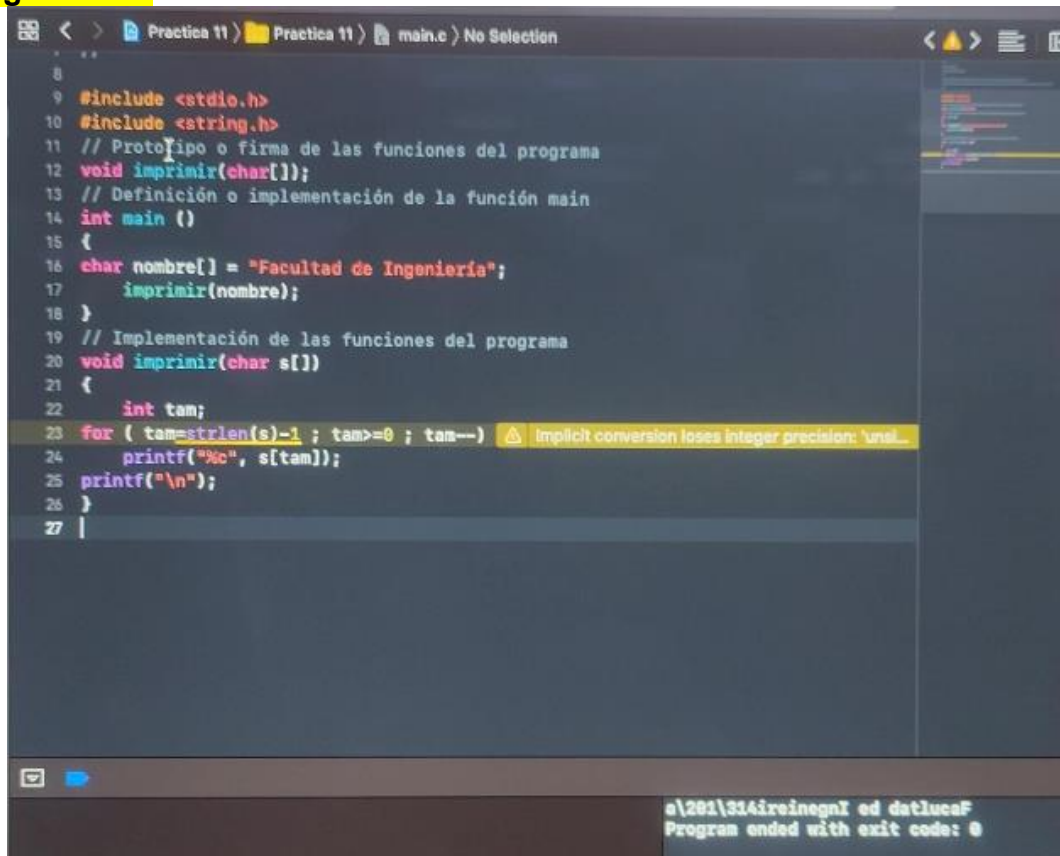
```
tipoValorRetorno nombre (parámetros)
{
    // bloque de código de la función
}
```

El nombre se refiere al identificador, una función puede recibir parámetros, los cuales son datos de entrada con los que trabajara la función, estos deben ser definidos dentro de los paréntesis, separados por comas e indicando su tipo de dato, puede ser entero, real, carácter o arreglo. Ejemplo:

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

El tipo de valor de retorno de una función indica el tipo de dato que va a regresar la función al terminar el bloque de código de esta a través de la sentencia return.

## Programa 1.c



```
8
9
10 #include <stdio.h>
11 #include <string.h>
12 // Prototipo o firma de las funciones del programa
13 void imprimir(char[]);
14 // Definición o implementación de la función main
15 int main ()
16 {
17     char nombre[] = "Facultad de Ingeniería";
18     imprimir(nombre);
19 }
20 // Implementación de las funciones del programa
21 void imprimir(char s[])
22 {
23     int tam;
24     for ( tam=strlen(s)-1 ; tam>=0 ; tam--)
25         printf("%c", s[tam]);
26     printf("\n");
27 }
```

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

Este fue el primer programa que se ejecutó en esta serie, en la parte de abajo se puede apreciar lo que nos imprimió, lo cual es “Facultad de Ingeniería” pero escrito al revés.

Este programa puede funcionar de dos maneras diferentes, la primera es declarando la firma hasta arriba y posteriormente a la principal, o colocando las funciones antes de llamar a la función principal.

```
Practica 11 > Practica 11 > main.c > Imprimir(a)
6 // Copyright © 2022 Avila Pineda Samuel David. 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 int main ()
15 {
16     char nombre[] = "Facultad de Ingeniería";
17     imprimir(nombre);
18 }
19 // Implementación de las funciones del programa
20 void imprimir(char s[])
21 {
22     int tam;
23     for ( tam=0 ; tam<strlen(s)-1 ; tam++)
24         printf("%c", s[tam]);
25     printf("\n");
26 }
27
```

Facultad de Ingeniería  
Program ended with exit code: 0

Como actividad extra nos dejó que modificáramos el programa de tal manera que “Facultad de Ingeniería” lo imprimiera bien.

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

#### **Programa 2.c**

```
#include <stdio.h>
10 void sumar();
11 int main() {
12     sumar(); // llamado de la función suma
13 }
14
15 void sumar() // función suma
16 {
17     int x=5, y=10, z; //variables locales
18     z=x+y;
19     printf("%i", z);
20 }
21
22
23
```

This function declaration is not a prototype

15Program ended with exit code: 0

Este segundo programa se modificaron las comillas y se declaró la firma antes del main.

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

15 Program ended with exit code: 0

Algo extra fue colocar un salto de línea en el renglón 19, con el fin de que no se viera tan amontonado lo que nos iba a imprimir.

### Programa 3.c

```
#include <stdio.h>

int resultado; //variable global

int main()
{
    multiplicar(); //llamado de la función multiplicar
    printf("%i",resultado);
    return 0;
}

void multiplicar() //función multiplicar
{
    resultado = 5 * 4;
    return 0;
}
```

```
9  #include <stdio.h>
10 void multiplicar();
11 int resultado; //variable global
12 int main()
13 {
14     multiplicar(); //llamado de la función multiplicar
15     printf("%i\n", resultado);
16     return 0;
17 }
18 void multiplicar() //función multiplicar
19 {
20     resultado = 5 * 4;
21 }
22
```

This function declaration is not a prototype

20  
Program ended with exit code: 0

En este tercer programa se hicieron unas pequeñas modificaciones al original, fueron cambiar las comillas y agregar un salto de línea como el ejercicio anterior, además le faltaba colocar después del `#include<stdio.h>` el `void multiplicar`. La profesora nos dijo que comentáramos la variable local, pero no había.

#### Programa 4.c

```
9
10 #include <stdio.h>
11 void incremento();
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     return 999; }
25 void incremento()
26 {
27     // La variable enteraLocal es local a la función incremento
28     int enteraLocal = 5;
29     enteraGlobal += 2;
30     printf("global(%i) + local(%i) = %d\n", enteraGlobal, enteraLocal,
31           enteraGlobal+enteraLocal);
32     //return 0; Esta línea de código no se debe colocar debido a que el tipo
33     de dato de retorno es void
34 }
```

```
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
```

La única modificación que se hizo en este programa fue comentar el return, lo que coloqué, porque fue lo que entendí de la maestra es que el return no se debía de colocar porque el tipo de dato de retorno es void.

### **Argumentos para la función main**

La firma de una función se compone de 3 elementos: el tipo del valor de retorno de la función, el nombre de la función y los parámetros que recibe la función.

La firma completa de la función main es:

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

La función main puede recibir como parámetro de entrada un arreglo de cadenas al ejecutar el programa

### **Programa 5.c**



```

#include <stdio.h>
#include <string.h>
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;
}

```

```

[Colombia27:~ fp20alu06$ gcc pro5.c -o pro5.out
[Colombia27:~ fp20alu06$ ./pro5.out
El programa no contiene argumentos.
[Colombia27:~ fp20alu06$ vi pro5.c
[Colombia27:~ fp20alu06$ ./pro5.out a b c d e
Los elementos del arreglo argv son:
argv[0] = ./pro5.out
argv[1] = a
argv[2] = b
argv[3] = c
argv[4] = d
argv[5] = e
[Colombia27:~ fp20alu06$ ./pro5.out S A M U E L 1 2
Los elementos del arreglo argv son:
argv[0] = ./pro5.out
argv[1] = S
argv[2] = A
argv[3] = M
argv[4] = U
argv[5] = E
argv[6] = L
argv[7] = 1
argv[8] = 2
Colombia27:~ fp20alu06$

```

El quinto programa se hizo en la terminal, ya que en xforce se nos iba a complicar un poco más, a la hora de ejecutar el programa debíamos de colocar variables separados por espacios, para que a la hora de que nos imprimiera nos apareciera de forma vertical.

### Estático

Sintaxis:

```

static tipoDato nombre;
static tipoValorRetorno nombre(parámetros);

```

Tanto la declaración de una variable como a la firma de una función solo se le agrega la palabra reservada static al inicio de estas.

Static es una variable que hace que esta pertenezca en memoria desde su creación



y durante toda la ejecución del programa, en otras palabras su valor se mantendrá hasta que el programa llegue a su fin.

### Programa 6.c

```
9 #include <stdio.h>
10 void llamarFuncion();
11 int main () {
12     for (int j=0 ; j < 5 ; j++) {
13         llamarFuncion();
14     }
15     void llamarFuncion() {
16         /* Solo la primera vez que se llame a esta función se creará y se le asignará el valor de 0 a la
17            variable estática numVeces */
18         static int numVeces = 0;
19         printf("Esta función se ha llamado %d veces.\n", ++numVeces);
20     }
21 }
```

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

### FuncEstatica.c

```
//##### funcEstatica.c #####
#include <stdio.h>

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

```

//##### calculadora.c #####
#include <stdio.h>

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));
}

```

Estos dos últimos se me complicaron, lo único que recuerdo con estos dos programas es que en el archivo xforce, en una carpeta se abrieron dos archivos y en uno se colocó la calculadora y en el otro la Estática.

```

9  int suma(int,int);
10 static int resta(int,int);
11 int producto(int,int);
12 static int cociente (int,int);
13 int suma (int a, int b) {
14     return a + b; }
15 static int resta (int a, int b) {
16     return a - b; }
17 int producto (int a, int b) {
18     return (int)(a*b); }
19 static int cociente (int a, int b) {
20     return (int)(a/b); }
21

```

Unused function 'resta'

Unused function 'cociente'

```

5 + 7 = 12
6 * 8 = 48
Program ended with exit code: 0

```

```

9  #include <stdio.h>
10 int suma(int,int);
11 //static int resta(int,int); int producto(int,int);
12 //static int cociente (int,int);
13 int main()
14 {
15     printf("5 + 7 = %i\n",suma(5,7));
16     //printf("9 - 77 = %d\n",resta(9,77));
17     printf("6 * 8 = %i\n",producto(6,8));
18     //printf("7 / 2 = %d\n",cociente(7,2));
19 }
20

```

Implicit declaration of function 'cociente'

```

5 + 7 = 12
6 * 8 = 48
Program ended with exit code: 0

```

```

9  #include "func_estatica.h"
10 //##### funcEstatica.c #####
11 #include <stdio.h>
12 int suma(int,int);
13 int producto(int,int);
14 int cociente (int,int);
15 int suma (int a, int b) {
16     return a + b; }
17 int producto (int a, int b) {
18     return (int)(a*b); }
19 int cociente (int a, int b) {
20     return (int)(a/b); }
21

```

```

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

```

```

10 #include <stdio.h>
11 int suma(int,int);
12 static int resta(int,int);
13 //static int resta(int,int);
14 int producto(int,int);
15 static int resta (int a, int b) {
16     return a - b; }
17
18 //static int cociente (int,int);
19 int main()
20 {
21     printf("5 + 7 = %i\n",suma(5,7));
22     printf("9 - 77 = %d\n",resta(9,77));
23     printf("6 * 8 = %i\n",producto(6,8));
24     printf("7 / 2 = %d\n",cociente(7,2));
25 }
26

```

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

```

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

```

## Tarea

Del ejercicio proporcionado en clase se debe realizar lo siguiente:

Completar ejercicio

Mediante prueba de escritorio dibujar la salida en el recuadro

Fragmentar el programa para emplear 3 funciones, una de ellas la función principal  
Del programa fragmentado generar su codificación.

SAMUEL DAVID

Compare 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  $n, k, j$ ;  
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 " $n$ "  
FINSI

PARA  $k=2$  HASTA  $k=n-1$  CON PASO 1 HACER  
PARA  $j=0$  HASTA  $j=n-k$  CON PASO 1 HACER  
ESCRIBIR " "  
FINPARA  
ESCRIBIR "\*\*\*"  
PARA  $j=1$  HASTA  $j=2*k-1$  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

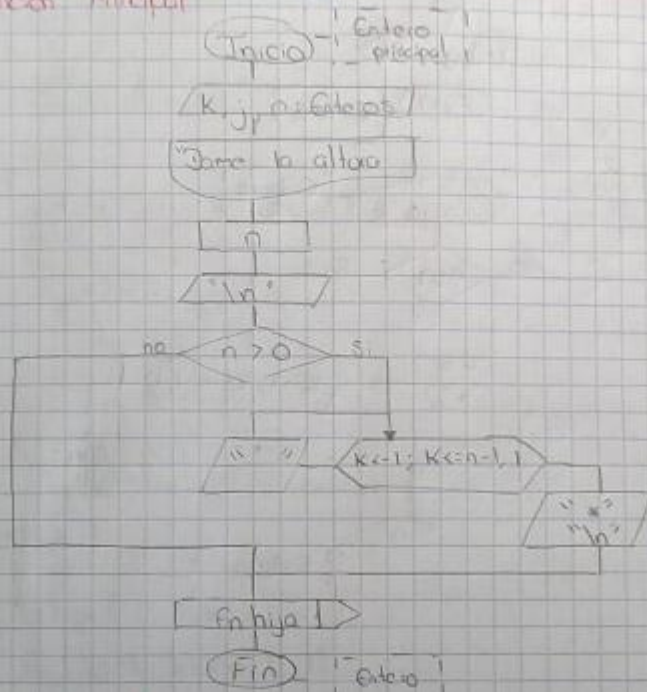
**FIN**

```
graph TD
    INICIO([INICIO]) --> P1[Dame la altura]
    P1 --> L1[n]
    L1 --> P2["n"]
    P2 --> D1{n=0}
    D1 -- SI --> P3[" "]
    P3 --> D2{k=1, k=n-1, 1}
    D2 --> P4[" "]
    P4 --> D3{n}
    D3 -- NO --> P5[" "]
    P5 --> D4{k=2, k=n-1, 1}
    D4 --> P6[" "]
    P6 --> D5{j=0, j=n-k, 1}
    D5 --> P7[" "]
    P7 --> D6{j=1, j=2*k-1, 1}
    D6 --> P8[" "]
    P8 --> P9["***"]
    P9 --> P10["n"]
    P10 --> D7{n > 1}
    D7 -- SI --> P11["***"]
    P11 --> D8{k=1, k=n-1, 1}
    D8 --> P12[" "]
    P12 --> P13["***"]
    P13 --> P14["n"]
    D7 -- NO --> FIN([FIN])
```

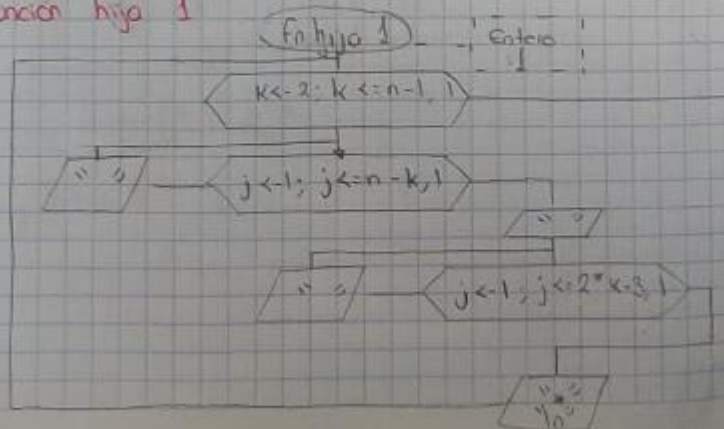
No me salió, no entendí

# Diagrama de Flujo

Función Principal

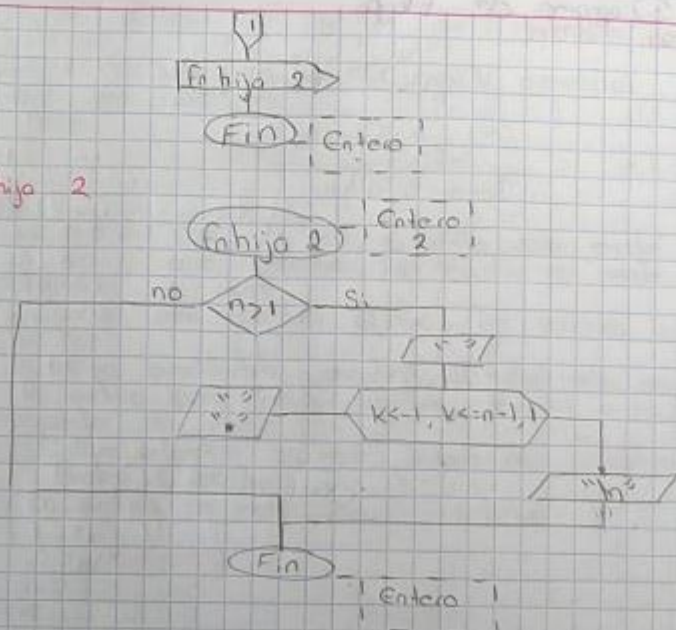


Función hijo 1





Función hijo 2



## Pseudocódigo

INICIO

Declarar  $k, j, n$  como enteros

ESCRIBIR "Dime k y n"

Leer  $n$

ESCRIBIR "\n"

Si  $(n > 0)$  Entonces

PARA ( $k \leftarrow 1; k \leftarrow n-1; 1$ )

ESCRIBIR "

FIN PARA

ESCRIBIR " "

ESCRIBIR "\n"

Fin Si

func\_hija1()

FIN

INICIO func\_hija1

Declarar  $func_hija1(int k, j)$  como entero

PARA ( $k \leftarrow 2; k \leftarrow n-1; 1$ )

PARA ( $j \leftarrow 1; j \leftarrow n-1; 1$ )

ESCRIBIR "

FIN PARA

ESCRIBIR " "

PARA ( $j \leftarrow 1; j \leftarrow 2 * k - 3; 1$ )

ESCRIBIR "

FIN PARA

ESCRIBIR " "

ESCRIBIR "\n"

FIN PARA

func\_hija2()

FIN func\_hija1

INICIO func\_hija2

Declarar  $func_hija2(int n, k)$  como entero

Si  $(n > 1)$  Entonces

ESCRIBIR "

PARA ( $k \leftarrow 1; k \leftarrow n-1; 1$ )

ESCRIBIR "

ESCRIBIR " "

FIN PARA

ESCRIBIR "\n"

Fin Si

FIN func\_hija2



## Codificación

```
#include <stdio.h>
int k, j, n;
int fahija1(int k, j);
int fahija2(int n, k);

int main
{
    printf("Dame la altura");
    scanf("%d", &n);
    printf("\n");
    if (n > 0)
    {
        for (k <= 1; k <= n-1; 1)
        {
            printf(" ");
        }
        printf("\n");
        printf("\n");
    }
    fahija1
}

int fahija1(int k, j)
{
    for (k <= 2; k <= n-1; 1)
    {
        for (j <= 1; j <= n-k; 1)
        {
            printf(" ");
        }
        printf("\n");
    }
    fahija2
}
```

```
int fahija2(int n, k);
if (n > 1)
{
    printf(" ");
    for (k <= 1; k <= n-1; 1)
    {
        printf(" ");
        printf("\n");
    }
}
```

## Conclusiones

Esta práctica al principio no se me hizo tan complicada, sin embargo, al final lo que se complicó fue lo de la calculadora y lo de estática, no sé porque, yo creo que fue debido a que hay que checar ambos programas y que ninguno tenga un error o algo diferente.

Aquí pudimos ver la importancia de las funciones, básicamente estas sirven para almacenar algo hasta al fin del programa, esto nos ayuda bastante, ya que así no escribimos todo de nuevo, mientras la función este guardada.

## Referencias

- <http://lcp02.fi-b.unam.mx/>
- El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991