	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): Karina García Morales

Asignatura: Fundamentos de Programación

Grupo: 22

No de Práctica(s): 11

Integrante(s): Edilberto Vicente Martínez

No. De lista o brigada: 50

Semestre: 2026-1

Fecha de entrega: 18 de noviembre de 2025

Observaciones:

CALIFICACIÓN: _____

Práctica 11: Funciones

Objetivo:

El alumnado elaborará programas en C donde la solución del problema se divida en funciones. Distinguirá lo que es el prototipo o firma de una función y la implementación de ella, asimismo manipulará parámetros tanto en la función principal como en otras.

Desarrollo:

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, el sistema inicia 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.

Funciones

La sintaxis básica para definir una función es la siguiente:

```
tipoValorRetorno 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, 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 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 ésta a través de la sentencia return. El valor del retorno puede ser entero, carácter o arreglo, o en un dado caso para el tipo de la función, se puede omitir la palabra return por la palabra reservada void (vacío o genérico).

Una declaración, prototipo o firma de una función tiene la siguiente sintaxis:

tipoValorRetorno nombre (parámetros);

La firma de una función está compuesta por tres elementos: el tipo del valor de retorno de la función, el nombre de la función y los parámetros que recibe la función; finaliza con punto y coma (;). Los nombres de los parámetros no necesariamente deben ser iguales a los que se encuentran en la definición de la función. Las funciones definidas en el programa no necesariamente deberán ser declaradas; esto dependerá de su ubicación en el código.

NOTA: strlen es una función que recibe como parámetro un arreglo de caracteres y regresa como valor de retorno un entero que indica la longitud de la cadena. La función se encuentra dentro de la biblioteca string.h, por eso se incluye ésta al principio del programa.

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. Las variables que se declaren dentro de cada función se conocen como variables locales (a cada función). Estas variables existen al momento de que la función es llamada y desaparecen cuando la función llega a su fin.

Las variables que se declaran fuera de cualquier función se llaman variables globales. Las variables globales existen durante la ejecución de todo el programa y pueden ser utilizadas por cualquier función.

Argumentos para la función main

La firma o prototipo de una función está compuesta por tres elementos: el tipo del valor de retorno de la función, el nombre de la función y los parámetros de la función. La función main también puede tener parámetros. Si se declaran parámetros en la función main deben ser dos: el primero se declara de tipo entero y el segundo se declara como un arreglo de cadenas; un ejemplo de la cabecera de la función main con parámetros es:

int main (int argc, char ** argv)

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). Para enviar argumentos a la función main, el programa se debe ejecutar de la siguiente manera, en la línea de comandos:

- En plataforma Linux/Unix ./nombrePrograma arg1 arg2 arg3 ...
- En plataforma Windows nombrePrograma.exe arg1 arg2 arg3 ...

Esto es, el nombre del programa seguido de los argumentos de entrada. De tal manera que los parámetros recibirán los siguientes valores, suponiendo los nombres de la cabecera de ejemplo mostrada anteriormente:

argc: número de cadenas contabilizadas en la línea de comandos, incluyendo el nombre del programa. argv: arreglo con las cadenas de la línea de comandos: argv[0] es el nombre del programa. argv[1] es la cadena de arg1 argv[2] es la cadena de arg2 y así sucesivamente.

Estático

El Lenguaje C permite definir elementos estáticos. La sintaxis para declarar elementos estáticos es la siguiente:

static tipoDato nombre;

static tipoValorRetorno nombre(parámetros);

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

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.

El atributo static en una función hace que esa función sea accesible solo dentro del mismo archivo, lo que impide que fuera de la unidad de compilación se pueda acceder a la función.

Tarea:

1.- ¿Qué es una función?

Es un bloque de código el cual sirve para realizar alguna tarea específica como un cálculo o una realización de una operación.

2.- Describe los 4 tipos de funciones:(función sin tipo de dato de retorno y sin parámetros de entrada, etc.)

Función con parámetros de entrada y tipo de dato de retorno:
int nombreFuncion(int a,int b)

Función con parámetros de entrada y sin tipo de dato de retorno:
nombreFuncion(int a,int b) void nombreFuncion(int a,int b)

Función sin parámetros de entrada y tipo de dato de retorno:
int nombreFuncion( )

Función sin parámetros de entrada y sin tipo de dato de retorno:
nombreFuncion( )
void nombreFuncion( )

3.- Sintaxis de la función:

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

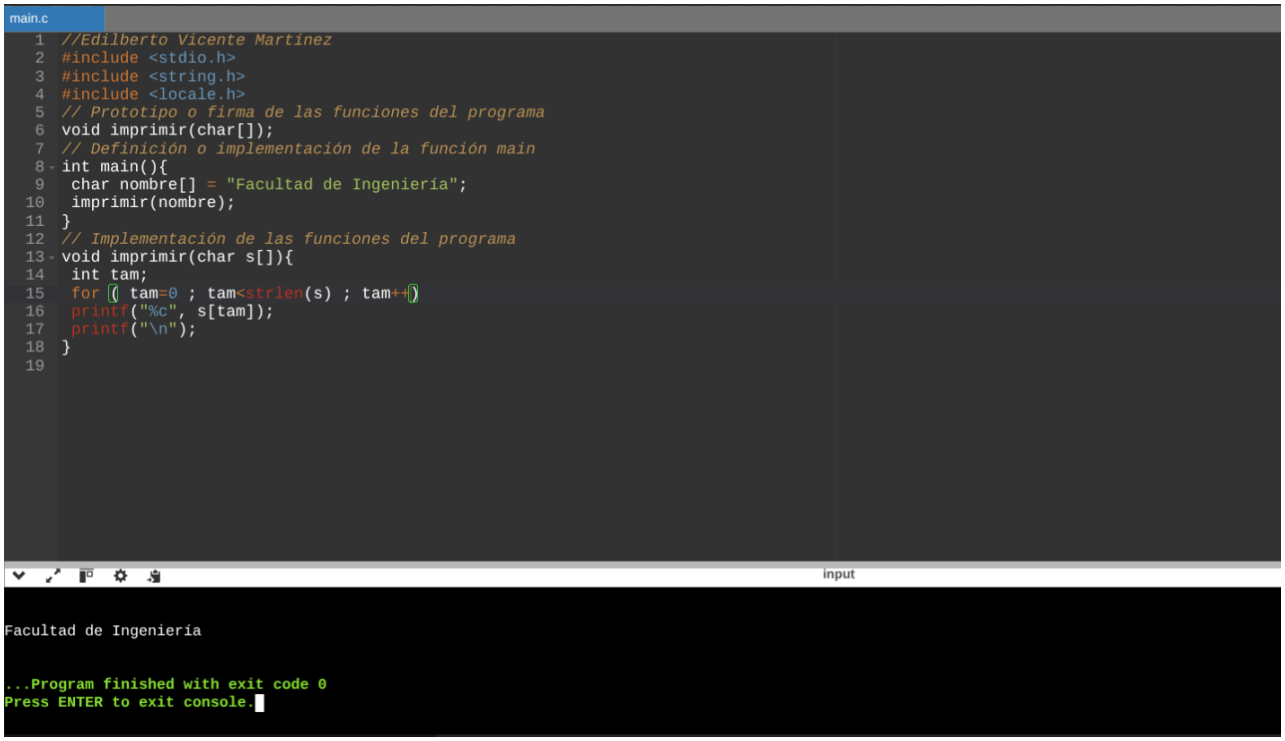
4.- ¿Qué es la firma de la función?

```
tipoValorRetorno nombre (parámetros);
```

En donde;

- **tipoValorRetorno:** El tipo de dato que la función devolverá al finalizar (por ejemplo, `int`, `float`, `void` si no devuelve nada).
- **Nombre:** El identificador único de la función.
- **Parámetros:** Los datos de entrada que la función puede recibir. Cada parámetro debe tener un tipo de dato y, opcionalmente, un nombre de variable que se usará dentro de la función.

5.- Modificar el programa 1.c para que imprima el texto correctamente y permitir acentos.



*Impresión del programa.

6.- Modifica el programa 2.c para que el usuario ingrese las variables "x" y "y"

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
void sumar(); // prototipo de La función
int main()
{
sumar(); // Llamado de la función suma
return 0;
}
void sumar() // Nuestra función suma
{
int x, y, z; // variables Locales
// Solicitar al usuario que ingrese Los valores DENTRO de La función
printf("Ingrese el valor de X: ");
scanf("%d", &x);

printf("Ingrese el valor de Y: ");
scanf("%d", &y);
z = x + y; // Realizar la suma
// Mostrar el resultado
printf("La suma de %d + %d = %d\n", x, y, z);
}
```

input

```
Ingrese el valor de X: 7
Ingrese el valor de Y: 8
La suma de 7 + 8 = 15


...Program finished with exit code 0
Press ENTER to exit console.
```

*Codificación e impresión del programa.

7.- Indica la diferencia entre la variable global y local

Las variables locales son declaradas dentro de una función, estas aparecen cuando la función es llamada y desaparecen cuando la función termina. Mientras que las variables globales son ejecutadas durante todo el programa y pueden ser utilizadas para cualquier función.

8. Ejecuta el programa 5.c con 5 argumentos e indica que es un argumento
Un argumento es un valor de entrada que ingresa cuando se llama a la función en el que esta se encuentra

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
#include <string.h>
int main(int argc, char** argv)
{
 if (argc == 1)
 {
 printf("El programa no contiene argumentos.\n");
 return 0;
 }
 printf("Los elementos del arreglo argv son:\n");
 for (int cont = 0 ; cont < argc ; cont++ ){
 printf("argv[%d] = %s\n", cont, argv[cont]);
 }
 return 0;
}
```

input

```
El programa no contiene argumentos.


...Program finished with exit code 0
Press ENTER to exit console.
```

*Impresión del programa (no pude poner los elementos en mi dispositivo).

9.- Ejecuta el programa 6.c e indica que hace una variable estática.

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
void llamarFuncion();
int main()
{
 for (int j=0 ; j < 5 ; j++)
 {
 llamarFuncion();
 }
}

void llamarFuncion()
{
 /* Solo la primera vez que se llame a esta función se creará y se le asignará
 el valor de 0 a la variable estática numVeces */
 static int numVeces = 0;
 printf("Esta función se ha llamado %d veces.\n",++numVeces);
}
```

input

```
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.
```

La variable estática hace que un valor permanezca con su mismo valor durante toda la ejecución del programa.

Fragmentar el programa para emplear 3 funciones, una de ellas la función principal (main). Del programa fragmentado generar su codificación.(Diagrama de flujo y pseudocódigo).

main.c

```c
//Edilberto Vicente Martinez
#include <stdio.h>

void imprimirCuerpo(int n);
void imprimirBase(int n);
int main() {
    int n;
    printf("Dame la altura: ");
    scanf("%d", &n);
    printf("\n");

    if (n > 0) {
        imprimirCuerpo(n);
        if (n > 1) {
            imprimirBase(n);
        }
    }

    return 0;
}
void imprimirCuerpo(int n) {
    int k, j;
    for (k = 1; k <= n - 1; k++) {
        printf(" ");
    }
    printf("* \n");
    for (k = 2; k <= n - 1; k++) {
        for (j = 1; j <= n - k; j++) {
            printf(" ");
        }
        printf("* ");
        for (j = 1; j <= 2 * k - 3; j++) {
            printf(" ");
        }
        printf("*\n");
    }
}
void imprimirBase(int n) {
```

```c
    scanf("%d", &n);
    printf("\n");

    if (n > 0) {
        imprimirCuerpo(n);
        if (n > 1) {
            imprimirBase(n);
        }
    }

    return 0;
}
void imprimirCuerpo(int n) {
    int k, j;
    for (k = 1; k <= n - 1; k++) {
        printf(" ");
    }
    printf("* \n");
    for (k = 2; k <= n - 1; k++) {
        for (j = 1; j <= n - k; j++) {
            printf(" ");
        }
        printf("* ");
        for (j = 1; j <= 2 * k - 3; j++) {
            printf(" ");
        }
        printf("*\n");
    }
}
void imprimirBase(int n) {
    int k;
    printf("* ");
    for (k = 1; k <= n - 1; k++)
    {
        printf(" *");
    }
    printf("\n");
}
```

main.c

```c
//Edilberto Vicente Martinez
#include <stdio.h>

void imprimirCuerpo(int n);
void imprimirBase(int n);
int main() {
    int n;
    printf("Dame la altura: ");
    scanf("%d", &n);
    printf("\n");

    if (n > 0) {
        imprimirCuerpo(n);
        if (n > 1) {
            imprimirBase(n);
        }
    }

    return 0;
}
void imprimirCuerpo(int n) {
    int k, j;
    for (k = 1; k <= n - 1; k++) {
        printf(" ");
    }
    printf("* \n");
    for (k = 2; k <= n - 1; k++) {
        for (j = 1; j <= n - k; j++) {
            printf(" ");
```

input

```
Dame la altura: 6

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

*Codificación y ejecución del programa.

Pseudocódigo

```
Funcion imprimirCuerpo(n: Entero)
    Variables k, j: Entero
    // Imprimir punta
    Para k <- 1 Hasta n-1 Hacer
        Escribir " "
    FinPara
    Escribir "* ", SaltoDeLinea

    // Imprimir centro hueco
    Para k <- 2 Hasta n-1 Hacer
        Para j <- 1 Hasta n-k Hacer
            Escribir " "
        FinPara
        Escribir "* "
        Para j <- 1 Hasta 2*k-3 Hacer
            Escribir " "
        FinPara
        Escribir "*", SaltoDeLinea
    FinPara
FinFuncion

Funcion imprimirBase(n: Entero)
    Variable k: Entero
    Escribir "* "
    Para k <- 1 Hasta n-1 Hacer
        Escribir " *"
    FinPara
    Escribir SaltoDeLinea
```

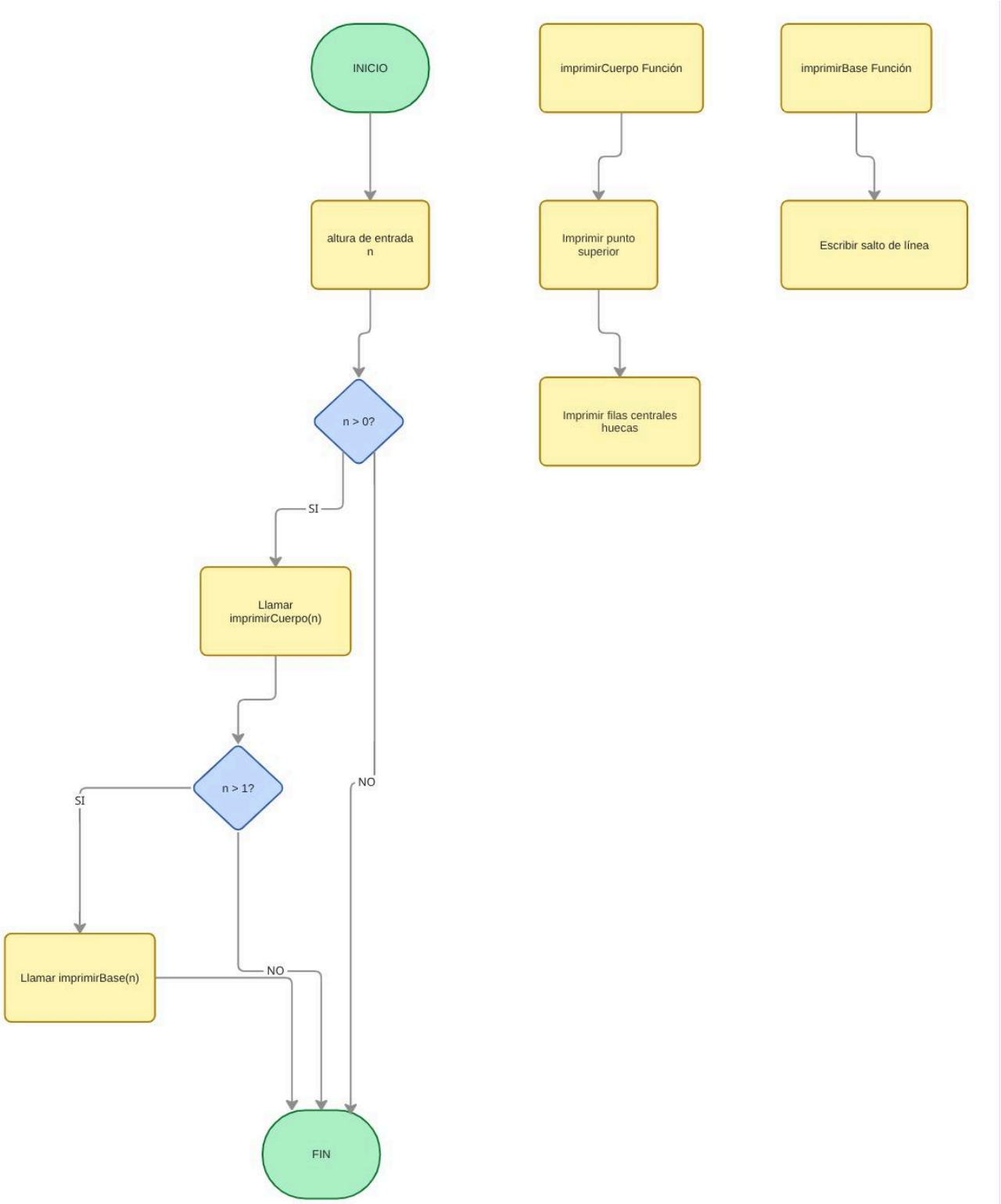
```
FinFuncion

// --- FUNCIÓN PRINCIPAL ---
Inicio Programa Principal
    Variable n: Entero
    Escribir "Dame la altura"
    Leer n

    Si n > 0 Entonces
        Llamar imprimirCuerpo(n)

        Si n > 1 Entonces
            Llamar imprimirBase(n)
        FinSi
    FinSi
Fin Programa
```

Diagrama de Flujo



Conclusión

Esta práctica fue muy interesante porque pude conocer el cómo podemos llegar a organizar nuestro programa y hacerlo mucho más práctico con la ayuda de las funciones y los archivos, además pude conocer qué tipo de palabras y variable están reservadas para las funciones y también pude conocer la diferencia entre una variable local y una variable global. En lo personal el implementar funciones es algo muy bueno, ya que te ayuda no solamente a organizar tu programa, sino que también nos puede ayudar a que la corrección de nuestro programa sea mejor.

Bibliografía

*Facultad de Ingeniería. (2025). Manual de prácticas del laboratorio de Fundamentos de Programación. Laboratorio de computación. Salas A y B. Universidad Nacional Autónoma de México. pp. 162-174. Recuperado el 18 de noviembre 2025 de http://lcp02.fi-b.unam.mx/

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