	**Carátula para entrega de prácticas**
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Karina García Morales

Asignatura: Fundamentos de programación

Grupo:

No. de Práctica: 10

Integrantes: Ever Ivan Rosales Gómez

No de Lista o brigade:

Semestre: 2023-2

Fecha de entrega: 31/Mayo/2023

Observaciones

CALIFICACIÓN

FUNCIONES

OBJETIVO

Elaborar programas en lenguaje C con ayuda de funciones, conocer el concepto de prototipo o firma de una función

DESARROLLO

Es importante saber el concepto de función para llevar a cabo esta práctica. Una función es una conjunto de instrucciones para llevar a cabo algo, por ejemplo; es como la máquina que va hacer el conjunto de operaciones dependiendo el programa, el anterior ejemplo se ve mucho en álgebra.

En programación supongamos que se debe hacer un programa de una calculadora que haga las 4 operaciones básicas (suma, resta, multiplicación y división). Pues este programa se puede usar funciones entonces cada operación será una función, cada una de las operaciones llevará a cabo un bloque de instrucciones con sus respectivas operaciones. Para posteriormente llamarlas en la función principal.

La manera de declarar una función es la siguiente siguiendo el ejemplo de la calculadora.

TIPO DE LA FUNCIÓN	NOMBRE DE LA FUNCIÓN	( )

int	suma	( ){
		instrucciones
		}

int	resta	( ){
		instrucciones
		}

int	multiplicación	( ){
		instrucciones
		}

int	división	( ){
		instrucciones
		}

Es de suma importancia mencionar que a lo largo del curso siempre usamos la función **main,** esta función es la principal para el lenguaje C.

PROTOTIPO DE UNA FUNCIÓN

Un prototipo permite declarar una función al inicio de un programa así como al declarar las variables con tu tipo de dato. Es recomendable hacer esto para que el programa no genere errores.

FIRMA DE UNA FUNCIÓN

Contiene 3 cosas de suma importancia

tipo del valor del retorno	Como ya se conocen los tipos de datos son: -enteros -reales -booleanos -carácter
nombre de la función	Este no debe tener un nombre específico, pero se puede recomendar ponerle dependiendo el propósito de la función.
parámetros	Los parámetros son los valores que la función espera recibir.

PROGRAMA 1

```
#include <stdio.h>
#include <string.h>
// Prototipo o firma de las funciones del programa
void imprimir(char[]);
// Implementación de las funciones del programa
void imprimir(char s[]){ int tam;
for ( tam=0; tam<=strlen(s) ; tam++)
printf("%c", s[tam]);
printf("\n");}
int main (){
char nombre[] = "Facultad de Ingeniería"; imprimir(nombre);

}
```

```
Bulgaria20:~ fp24alu35$ gcc prog1.c -o prog1.out
Bulgaria20:~ fp24alu35$ ./prog1.out
Facultad de Ingeniería
Bulgaria20:~ fp24alu35$
```

En este ejercicio tiene como propósito imprimir “Facultad de Ingeniería”, pero la primera vez que se compila y ejecuta la letras vienen al revés para cambiar esto es específicamente en el ciclo for donde la palabra (tam que es el nombre de la variable) debe ir en incremento. Algo importante es mencionar el **strlen** que es una **función** que recibe **parámetros de un carácter retornando un valor entero indicando la longitud de la cadena.**

PROGRAMA 2

```
#include <stdio.h>
//int x=5, y=10,z;
void sumar ();

int main() {
sumar(); // llamado de la función suma
}

void sumar() // función suma
{
    int x=5, y=10,z;// variables locales
    z=x+y;
    printf("%i",z);
}
```

```
Bulgaria20:~ fp24alu35$ gcc prog1.c -o prog1.out
Bulgaria20:~ fp24alu35$ ./prog1.out
15Bulgaria20:~ fp24alu35$
```

En este programa se encuentra una función con tipo de dato void en donde contiene un bloque de instrucciones como propósito sumar las variables “x” y “y” con valores 5 y 15 respectivamente. Para que esta función se realice se debe incluir en la función principal y hacer el **llamado a la función.**

PROGRAMA 3

```c
#include <stdio.h>
int resultado; //variable global

int multiplicar (); //firma de funcion
int main() {
   multiplicar(); //llamado de la función multiplicar
   printf("%i",resultado);
   //return 0;
}
int multiplicar() //función multiplicar
{
//int resultado;
resultado = 5 * 4;
return 0;
}
```

```
Bulgaria20:~ fp24alu35$ gcc prog1.c -o prog1.out
Bulgaria20:~ fp24alu35$ ./prog1.out
20Bulgaria20:~ fp24alu35$
```

Al igual que el programa 2, se incluye una función llamada multiplicar, dentro de esta tienen una variable llamada resultado que tiene como resultado la multiplicación de 5 y 4. Esta función se pondrá en la función principal para que se imprima así como se observa en la pantalla. Solamente que al final faltó anexar el printf(“\n”); para que el nombre de usuario se pusiera abajo.

VARIABLE GLOBALES Y LOCALES

Las variables globales son las que se ponen debajo de las librerías con el fin de ser utilizadas a lo largo de todo el código, por lo que es importante manejarlas de manera adecuada para que no se confundan con funciones. Por otro lado, las variables locales son aquellas que están dentro de una función y solo van a funcionar en esa función.

PROGRAMA 4

```
#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;
}
~
```

```
Bulgaria20:~ fp24alu35$ gcc prog2.c -o prog2.out
Bulgaria20:~ fp24alu35$ ./prog2.out
El programa no contiene argumentos.
Bulgaria20:~ fp24alu35$ ./prog2.out qwer rty 1 2 wer
Los elementos del arreglo argv son:
argv[0] = ./prog2.out
argv[1] = qwer
argv[2] = rty
argv[3] = 1
argv[4] = 2
argv[5] = wer
Bulgaria20:~ fp24alu35$
```

Las variables estáticas hacen que permanezcan en memoria desde su creación y durante toda la ejecución del programa. Se declara de la siguiente manera:

STATIC	**TIPO DE DATO**	**NOMBRE**	**( );**
static	int	var1	( );

PROGRAMA 5

```
#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 */
 int numVeces = 0;
printf("Esta función se ha llamado %d veces.\n",++numVeces);
}
~
```

```
Bulgaria20:~ fp24alu35$ vi prog2.c
Bulgaria20:~ fp24alu35$ gcc prog2.c -o prog2.out
Bulgaria20:~ fp24alu35$ ./prog2.out
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.
Bulgaria20:~ fp24alu35$ vi prog2.c
Bulgaria20:~ fp24alu35$ gcc prog2.c -o prog2.out
Bulgaria20:~ fp24alu35$ ./prog2.out
Esta función se ha llamado 1 veces.
Esta función se ha llamado 1 veces.
Esta función se ha llamado 1 veces.
Esta función se ha llamado 1 veces.
Esta función se ha llamado 1 veces.
Bulgaria20:~ fp24alu35$
```

En este ejercicio se utilizaron variables estáticas entonces el valor será el mismo a lo largo del código.

PROGRAMA 6

A continuación se muestran se muestran dos programas que se van a ejecutar juntos en la compilación y ejecución

```
//############## 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 #############
#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));
}
~
```

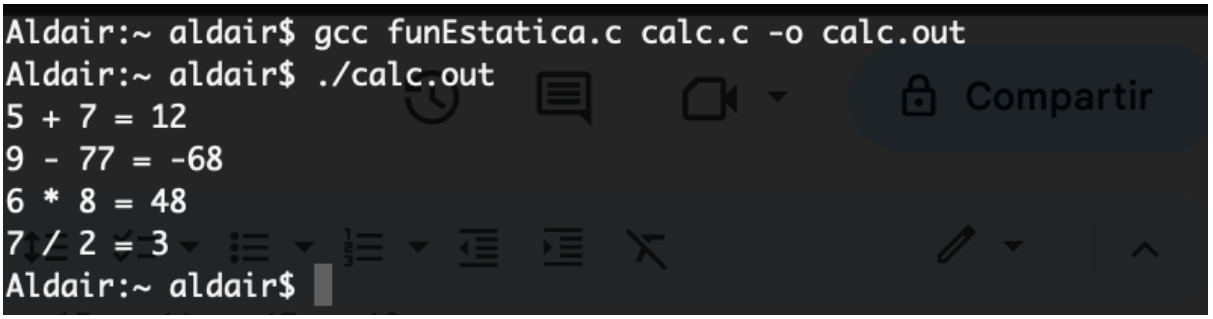
```
Aldair:~ aldair$ gcc funEstatica.c calc.c -o calc.out
Aldair:~ aldair$ ./calc.out
 5 + 7 = 12
 6 * 8 = 48
Aldair:~ aldair$
```

Como se muestra en el programa de nombre calc.c contiene las cuatro operaciones básicas y al juntarlos y compilar solo se muestran la suma y multiplicación

TAREA

```
//############## funcEstatica.c ##############
#include <stdio.h>
int suma(int,int); //funcion suma
int resta (int,int); //funcion resta
int cociente (int,int); //funcion cociente
int producto (int, int);

//int producto(int,int);
//static int cociente (int,int);

int suma (int a, int b)
{
 return a + b;
}

int resta (int a, int b)
{
 return a - b;
}

int producto (int a, int b)
{
 return (int)(a*b);
}

int cociente (int a, int b)
{
 return (int)(a/b);
}
~
```

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

int suma(int,int);
int resta (int,int);
int cociente(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));
}
~
```



Como se puede observar en este ejercicio se tienen dos programas, unos de ellos hace las 4 operaciones básicas y por otro lado se encuentra el programa llamado calculadora que tiene contiene a la función principal main donde se llamarán a las funciones (suma, resta, producto y residuo) para la compilación y ejecución de estos

CONCLUSIÓN

Con esta práctica se pone claro el propósito de la funciones que para mi la ventaja es que puedes llevar a cabo muchas instrucciones dentro de ella y fácilmente la puedes llamar en la función principal pues considero que la forma de en que se compila es la siguiente:

main(funcion1(funcion2 ()))

También es de suma importancia ya que en proyectos futuros serán códigos más complejos con muchas líneas de código.

REFERENCIAS

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