	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Marco Antonio Martinez Quintana
Asignatura:	FUNDAMENTOS DE PROGRAMACION
Grupo:	3
No de Práctica(s):	12
Integrante(s):	Andrik Uriel Reyes Roque
No. de Equipo de cómputo empleado:	No aplica
No. de Lista o Brigada:	Fp03alu39
Semestre:	1
Fecha de entrega:	22/01/2021
Observaciones:	

CALIFICACIÓN: _____

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.

Introducción

Una función es un mini programa dentro un programa. Las funciones contienen varias sentencias bajo un solo nombre que un programa puede utilizar una o más veces para ejecutar dichas sentencias. Las funciones ahorran espacio, reduciendo repeticiones y haciendo más fácil la programación, proporcionando un medio de dividir un proyecto grande en módulos pequeños más manejables. Las funciones existen de modo autónomo: cada una tiene su ámbito. Como ya conoce cada programa tiene al menos una función main (); sin embargo, cada programa C consta de muchas funciones en lugar de una función main () grande. La división del código en funciones hace que las mismas se puedan reutilizar en su programa y en otros programas. Después de que escriba, pruebe y depure su función, se puede utilizan nuevamente una y otra vez para reutilizar una función dentro de su programa, solo se necesita llamar a la función.

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

Código (funciones)

```c
#include <stdio.h>
#include <string.h>
/*
 Este programa contiene dos funciones: la función main y
imprimir. La función main manda llamar a la función impri
imprimir recibe como parámetro un arreglo de caracteres y
inicio imprimiendo cada carácter del arreglo.
*/
// Prototipo o firma de las funciones del programa
void imprimir(char[]);
// Definición o implementación de la función main
int main (){
 char nombre[] = "Facultad de Ingenieria";
 imprimir(nombre);
}
// Implementación de las funciones del programa
void imprimir(char s[]){
 int tam;
 for ( tam=strlen(s)-1 ; tam>=0 ; tam--)
 printf("%c", s[tam]);
 printf("\n");
}
```

```
C:\Users\PC\Desktop\Lenguaje C\Ejemplo>Funciones.exe
aireinegnI ed datlucaF

C:\Users\PC\Desktop\Lenguaje C\Ejemplo>
```

Código (Ámbito de las variables)

```c
#include <stdio.h>
/*
 Este programa contiene dos funciones: la función main y
función main manda llamar a la función incremento dentro
incremento aumenta el valor de la variable enteraGlobal c
*/
void incremento();
// La variable enteraGlobal es vista por todas
// las funciones (main e incremento)
int enteraGlobal = 0;
int main(){
 // La variable cont es local a la función main
 for (int cont=0 ; cont<5 ; cont++){
 incremento();
 }

 return 999;
}
void incremento(){
 // La variable enteraLocal es local a la función increme
 int enteraLocal = 5;
 enteraGlobal += 2;
 printf("global(%i) + local(%i) = %d\n",enteraGlobal, ent
enteraGlobal+enteraLocal);
}
```

```
C:\Users\PC\Desktop\Lenguaje C\Ejemplo>gcc Funciones.c -o Funciones.e

C:\Users\PC\Desktop\Lenguaje C\Ejemplo> Funciones.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\PC\Desktop\Lenguaje C\Ejemplo>
```

Código (argumentos función main)

```c
#include <stdio.h>
#include <string.h>
/*
 Este programa permite manejar los argumentos enviados
*/
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;
}
```

```
C:\Users\PC\Desktop\Lenguaje C\Ejemplo> Funciones.exe
El programa no contiene argumentos.

C:\Users\PC\Desktop\Lenguaje C\Ejemplo>
```

Estático

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

void llamarFuncion()
{
    static int numVeces = 0;
 printf("Esta función se ha llamado %d veces.\n",++numVeces);
}
```

```
C:\Users\PC\Desktop\Lenguaje C\Ejemplo>Funciones.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\PC\Desktop\Lenguaje C\Ejemplo>
```

Código (función estática)

```c
//############## funcEstatica.c ##############
#include <stdio.h>
/*
Este programa contiene las funciones de una calculadora básica: suma, resta, producto y
cociente.
*/
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>
/*
Este programa contiene el método principal, el cual invoca a las funciones
del archivo funcEstatica.c.
*/
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));
}
```

```
C:\Users\PC\Desktop\Lenguaje C\Ejemplo> Funciones.exe
5 + 7 = 12
6 * 8 = 48

C:\Users\PC\Desktop\Lenguaje C\Ejemplo>
```

Conclusión:

Con esta práctica aprendí que no solamente se le daba uso a la función main para la ejecución de un programa como habíamos estado viendo con anterioridad, sino que estas funciones las puedes ejecutar y a su vez estas tienen la posibilidad dentro del Código estar manejando varios datos y tener otras funciones dentro a las cuales pueden “llamar” a otras funciones y ejecutarse. Tanto el uso de nuevos conceptos como el void y int (con su respectivo nombre) te muestran que no siempre el retorno será 0 y dependiendo el tipo de dato que se utilice, este te regresará un valor o en su caso un vacío.

Bibliografía

Joyanes, Aguilar, Luis, and Martínez, Ignacio Zahonero. Programación en C, C++, Java y UML (2a. ed.), McGraw-Hill Interamericana, 2014. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/bibliodgbmhe/detail.action?docID=3225314.