	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: M.I. Marco Antonio Martínez Quintana

Asignatura: Fundamentos de Programación

Grupo: 3 Bloque: 136

No de Práctica(s): 12: Funciones

Integrante(s): Carranza Ochoa José David

No. de Equipo de cómputo empleado: No aplica

No. de Lista o Brigada: 6

Semestre: 2021-1

Fecha de entrega: 25/01/2021

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.

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.

Introducción

Como ya se mencionó, 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, se inicia con 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.

Desarrollo.

Funciones

La sintaxis básica para definir una función es la siguiente:

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

La función cuenta con parámetros a los cuales llama el programa, dichos parámetros serán ejecutados y trabajarán de acuerdo al bloque de instrucciones al cual se definan; de la misma forma se pueden agregar más de un parámetro dentro del bloque de instrucciones indicando el tipo de dato solicitado.

```
valorRetorno nombre (tipoDato nom1, tipoDato nom2, tipoDato nom3…)
```

La firma en una función cuenta con el nombre de la misma, los parámetros y los valores de retorno que pueda ofrecer, cabe destacar que no necesariamente las funciones pueden estar declaradas en el bloque inicial.

Ejemplo: código en el cuál se describen dos funciones

```c
#include <stdio.h>
#include <string.h>
/*
 Este 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.
*/
// Prototipo o firma de las funciones del programa
void imprimir(char[]);
// Definición o implementación de la función main
int main (){
 char nombre[] = "Facultad de Ingeniería";
 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");
}
```

```
D:\Programas\Dev-Cpp\ConsolePauser.exe
aÝreinegnI ed datlucaF

--------------------------------
Process exited with return value 10
Press any key to continue . . .
```

Ámbito o alcance de las variables

Es de suma importancia reconocer el tiempo de vida con el que cuentan las variables al momento de ser declaradas en la función principal, ya que se cuentan con variables locales y variables globales; para cada caso se tienen características especiales al momento de ser llamadas.

- Variables locales: se declaran dentro de cada función y desaparecen cuando la función llega a su fin
- Variables globales: se declaran fuera de cualquier función y su valor puede ser ocupado en cualquier momento del programa

```c
#include <stdio.h>
/*
 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.
*/
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 incremento
 int enteraLocal = 5;
 enteraGlobal += 2;
 printf("global(%i) + local(%i) = %d\n",enteraGlobal, enteraLocal,
enteraGlobal+enteraLocal);
}
```

```
D:\USUARIO\DOCUMENTOS_MECANICO\CompartidaU-W\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

--------------------------------
Process exited after 0.02612 seconds with return value 999
Presione una tecla para continuar . . .
```

Argumentos para la función main

Retomando las firmas de los códigos encontramos que las firmas están explicitas en el cuerpo del programa y que la firma completa de la función main es:

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

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

```
D:\USUARIO\DOCUMENTOS_MECANICO\CompartidaU-W\funciones.exe
El programa no contiene argumentos.

--------------------------------
Process exited after 0.08868 seconds with return value 88
Presione una tecla para continuar . . .
```

Estático

Existen elementos estáticos quienes permanecen en la memoria desde su creación hasta la finalización del programa, estos solo pueden accederse desde el mismo archivo, imposibilitando su extracción fuera del programa

static tipoDato nombre; static valorRetorno nombre(parámetros);

```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);
}
```

```
D:\USUARIO\DOCUMENTOS_MECANICO\CompartidaU-W\funciones.exe
Esta funci¾n se ha llamado 1 veces.
Esta funci¾n se ha llamado 2 veces.
Esta funci¾n se ha llamado 3 veces.
Esta funci¾n se ha llamado 4 veces.
Esta funci¾n se ha llamado 5 veces.

--------------------------------
Process exited after 0.02708 seconds with return value 0
Presione una tecla para continuar . . .
```

Teniendo la variable estática almacenada en la memoria ocasiona que al ser llamada por segunda vez está ya no se genere de nuevo, si no que retoma los valores con los que ya cuenta guardando su valor.

Ejemplo: dos archivos representados a la vez

```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));
}
```

```
D:\USUARIO\DOCUMENTOS_MECANICO\CompartidaU-W\funciones.exe
5 + 7 = 12
6 * 8 = 48

--------------------------------
Process exited after 0.02442 seconds with return value 0
Presione una tecla para continuar . . .
```

En dicho código coexisten dos archivos a la vez, la suma y el producto pertenecen al mismo archivo generando que se efectué el proceso solicitado; en cambio, al eliminar los comentarios se genera un error debido a las condiciones de las funciones estáticas ya que no pueden ser accedidas desde otro archivo externo.

Conclusiones

La división por secciones en los códigos es muy útil al momento de codificar algoritmos, contando con herramientas que dividan la función por partes es posible solucionar de manera más eficiente problemas a los que se exponga un programador. El parámetro “static” me resultó bastante interesante por sus características y restricciones que cuenta, reconozco que su implementación es indispensable en diversos eventos así que es necesario comprenderlo correctamente.

Finalmente, aprendí la sección “main” de la función, así como la firma que esta implementa en toda la sintaxis del software.

Bibliografía

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