



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Marco Antonio Martínez Quintana

Asignatura: Fundamentos de Programación

Grupo: 3

No de Práctica(s): 12

Integrante(s): Ramírez Pérez Daniela Itzel

*No. de Equipo de
cómputo empleado:* No aplica

No. de Lista o Brigada: 36

Semestre: 1er

Fecha de entrega: Viernes 22 de Diciembre

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

Un programa en lenguaje C consiste en una o más funciones. con el fin de dividir las tareas y que sea más eficiente el código.

En lenguaje C la función principal se llama *main* con las funciones dentro siendo llevadas a cabo.

Funciones

La sintaxis básica para definir una función es la siguiente:

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

Una función puede recibir parámetros de entrada dentro de paréntesis con los cuales trabajará, separados por comas e indicando su tipo de dato, de la siguiente forma:

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

El valor de retorno de una función indica el tipo de dato que va a regresar la función al terminar el bloque de código de la misma (entero, real, carácter o arreglo), o el elemento vacío (void).

El compilador C revisa que las funciones estén definidas o declaradas antes de ser invocada. Una declaración, prototipo o firma de una función tiene la siguiente sintaxis:

valorRetorno nombre (parámetros);

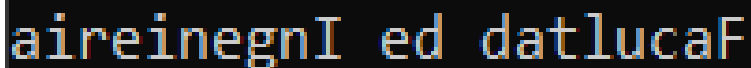
Compuesta por tres elementos:

- Nombre de la función
- Parámetros que recibe la función
- Valor de retorno de la función

Código (funciones)

```
#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 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");
}
```

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: Se declaran dentro de cada función en el momento en el que la función es llamada desaparecen.

```
void sumar() {
    int x;
    // ámbito de la variable x
}
```

- Variables globales: se declaran fuera de cualquier función existen durante la ejecución de todo el programa y pueden ser utilizadas por cualquier función

Código (Ámbito de las variables)

```
#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);
}
```

```
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
```

Argumentos para la función main

La función main también puede recibir parámetros y ellos deben ser enviados al ejecutar el programa. 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. 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 parámetros, el programa se debe ejecutar de la siguiente manera:

- En plataforma Windows

```
nombrePrograma.exe arg1 arg2 arg3 ...
```

Esto es, el nombre del programa seguido de los argumentos de entrada. Estos argumentos son leídos como cadenas de caracteres dentro del argument vector, donde en la posición 0 se encuentra el nombre del programa, en la posición 1 el primer argumento, en la posición 2 el segundo argumento y así sucesivamente.

Código (argumentos función main)

```
#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;
}
```

```
El programa no contiene argumentos
```

Estático

Lenguaje C permite definir elementos estáticos de la siguiente manera:

```
static tipoDato nombre;  
static valorRetorno nombre(parámetros);
```

El atributo static en una variable hace que ésta permanezca en memoria desde su creación y durante toda la ejecución del programa solo disponible en el mismo archivo.

Código (variable estática)

```
#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 funci%cn se ha llamado %d veces.\n",162,++numVeces);  
}
```

```
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.
```

Una vez declarada una variable estática, esta permanece en memoria a lo largo de la ejecución del programa, por lo tanto, la segunda vez que se llama a la función ya no se vuelve a crear la variable, si no que se utiliza la que está en la memoria y por eso conserva su valor.

Código (función estática)

Este ejemplo consta de dos archivos: funcEstatica.c y calculadora.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));
}
```

	$5 + 7 = 12$ $6 * 8 = 48$	
--	------------------------------	--

Conclusion

Con la practica hecha podemos ver todas las diferentes tipos de funciones que podemos declarar al principio de nuestro código para que tengan diferentes parámetros, las variables de nuestra elección, a lo largo de nuestro proyecto y también observamos como dos diferentes archivos C podemos compilarlos juntos para así podamos invocar variables de distintos archivos.

Referencias

- Nakayama Cervantes, A., Castañeda Perdomo, M., Solano Gálvez, J. A., García Cano, E. E., Sandoval Montaña, L., & Arteaga Ricci, T. I. (2018, 6 abril). Manual de prácticas del laboratorio de Fundamentos de programación (p.135-147). Laboratorio de Computación Salas A y B. http://odin.fib.unam.mx/salac/practicasFP/MADO-17_FP.pdf