



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: M.I. MARCO ANTONIO MARTINEZ QUINTANA

Asignatura: FUNDAMENTOS DE PROGRAMACIÓN

Grupo: 3

No de Práctica(s): 12

Integrante(s): CARRILLO CERVANTES IVETTE ALEJANDRA

*No. de Equipo de
cómputo empleado:* NO APLICA

No. de Lista o Brigada: 7

Semestre: PRIMER SEMESTRE

Fecha de entrega: 18 ENERO 2021

Observaciones:

CALIFICACIÓN: _____

Funciones

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

Funciones

La sintaxis básica para definir una función es la siguiente:

```
valorRetorno 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 de entrada, 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 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. El valor de retorno puede ser cualquiera de los tipos de datos vistos hasta el momento (entero, real, carácter o arreglo), aunque también se puede regresar el elemento vacío (void).

El compilador C revisa que las funciones estén definidas o declaradas antes de ser invocadas. Por lo que una buena práctica es declarar todas las funciones al inicio del programa. Una declaración, prototipo o firma de una función tiene la siguiente sintaxis:

valorRetorno nombre (parámetros);

La firma de una función está compuesta por tres elementos: el nombre de la función, los parámetros que recibe la función y el valor de retorno de 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.

funciones.c

```
File Edit Selection Find View Goto Tools Project Preferences Help
Funciones.c x RecetaMedica.c x
1 #include <stdio.h>
2 #include <string.h>
3 /*
4  Este programa contiene dos funciones: la función main y la
5  imprimir. La función main manda llamar a la función imprimir
6  imprimir recibe como parámetro un arreglo de caracteres y
7  inicio imprimiendo cada carácter del arreglo.
8  */
9  // Prototipo o firma de las funciones del programa
10 void imprimir(char[]);
11 // Definición o implementación de la función main
12 int main () {
13     char nombre[] = "Facultad de Ingeniería";
14     imprimir(nombre);
15 }
16 // Implementación de las funciones del programa
17 void imprimir(char s[]){
18     int tam;
19     for ( tam=strlen(s)-1 ; tam>=0 ; tam-- )
20         printf("%c", s[tam]);
21     printf("\n");
22 }
23
```

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. 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.

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

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.

```
#include <stdio.h>
int resultado;
void multiplicar() {
    resultado = 5 * 4;
}
```

AmbitoDeLasVariables.c

```
File Edit Selection Find View Goto Tools Project Preferences Help
Funciones.c x AmbitoDeLasVariables.c x
1 #include <stdio.h>
2 /*
3  Este programa contiene dos funciones: la función main y la función incremento. La
4  función main manda llamar a la función incremento dentro de un ciclo for. La función
5  incremento aumenta el valor de la variable enteraGlobal cada vez que es invocada.
6  */
7  void incremento();
8  // La variable enteraGlobal es vista por todas
9  // las funciones (main e incremento)
10 int enteraGlobal = 0;
11 int main(){
12     // La variable cont es local a la función main
13     for (int cont=0 ; cont<5 ; cont++){
14         incremento();
15     }
16     return 999;
17 }
18 void incremento(){
19     // La variable enteraLocal es local a la función incremento
20     int enteraLocal = 5;
21     enteraGlobal += 2;
22     printf("global(%i) + local(%i) = %d\n", enteraGlobal, enteraLocal,
23           enteraGlobal+enteraLocal);
24 }
```

Argumentos para la función main

Como se mencionó anteriormente, la firma de una función está compuesta por tres elementos: el nombre de la función, los parámetros que recibe la función y el valor de retorno de la función.

La función main también puede recibir parámetros. Debido a que la función main es la primera que se ejecuta en un programa, los parámetros de la función hay que enviarlos 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:

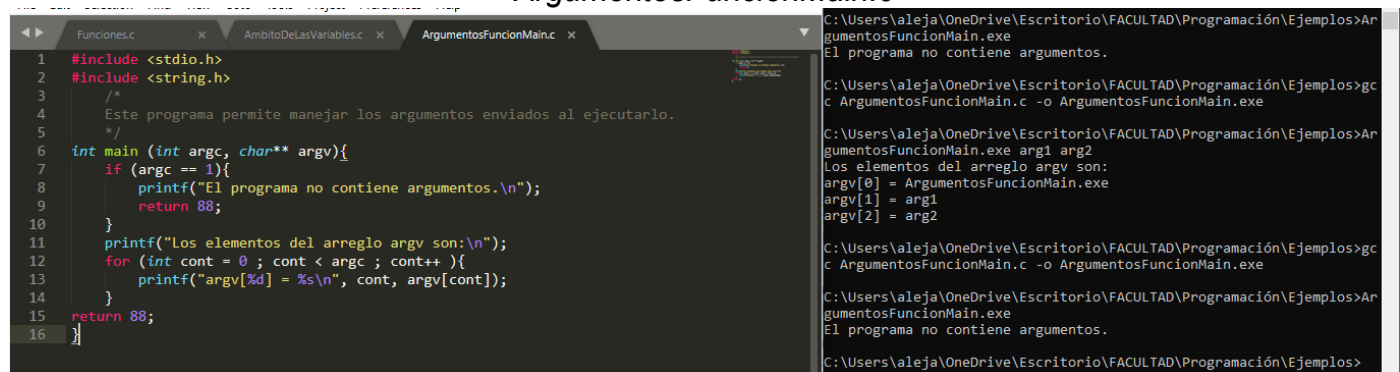
□ En plataforma Linux/Unix

`./nombrePrograma arg1 arg2 arg3 ...`

□ En plataforma Windows

`nombrePrograma.exe arg1 arg2 arg3 ...`

ArgumentosFuncionMain.c



```
1 #include <stdio.h>
2 #include <string.h>
3 /*
4  * Este programa permite manejar los argumentos enviados al ejecutarlo.
5  */
6 int main (int argc, char** argv){
7     if (argc == 1){
8         printf("El programa no contiene argumentos.\n");
9         return 88;
10    }
11    printf("Los elementos del arreglo argv son:\n");
12    for (int cont = 0 ; cont < argc ; cont++){
13        printf("argv[%d] = %s\n", cont, argv[cont]);
14    }
15    return 88;
16 }
```

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>Ar
gumentosFuncionMain.exe
El programa no contiene argumentos.

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>gc
c ArgumentosFuncionMain.c -o ArgumentosFuncionMain.exe

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>Ar
gumentosFuncionMain.exe arg1 arg2
Los elementos del arreglo argv son:
argv[0] = ArgumentosFuncionMain.exe
argv[1] = arg1
argv[2] = arg2

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>gc
c ArgumentosFuncionMain.c -o ArgumentosFuncionMain.exe

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>Ar
gumentosFuncionMain.exe
El programa no contiene argumentos.

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>
```

Estático

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

`static tipoDato nombre;`

`static valorRetorno 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 las mismas.

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.

VariableEstatica.c

```
File Edit Selection Find View Goto Tools Project Preferences Help
Funciones.c x AmbitoDeLasVariables.c x ArgumentosFuncionMain.c x VariableEstatica.c x
1 #include <stdio.h>
2 /*
3  Este programa contiene dos funciones: la función main y la función
4  llamarFuncion. La función main manda llamar a la función llamarFuncion dentro
5  de un ciclo for. La función llamarFuncion crea una variable estática e imprime
6  su valor.
7  */
8 void llamarFuncion();
9 int main () {
10     for (int j=0; j < 5; j++){
11         llamarFuncion();
12     }
13 }
14 void llamarFuncion(){
15     static int numVeces = 0;
16     printf("Esta función se ha llamado %d veces.\n", 162, ++numVeces);
17 }
```

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>gcc VariableEstatica.c -o VariableEstatica.exe
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>VariableEstatica.exe
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.
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>
```

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.

FunEstatica.c

```
File Edit Selection Find View Goto Tools Project Preferences Help
FunEstatica.c x Calculadora.c x
1 //##### funcEstatica.c #####
2 #include <stdio.h>
3 /*
4  Este programa contiene las funciones de una calculadora básica: suma,
5  resta, producto y cociente.
6  */
7 int suma(int,int);
8 static int resta(int,int);
9 int producto(int,int);
10 static int cociente (int,int);
11 int suma (int a, int b){
12     return a + b;
13 }
14 static int resta (int a, int b){
15     return a - b;
16 }
17 int producto (int a, int b){
18     return (int)(a*b);
19 }
20 static int cociente (int a, int b){
21     return (int)(a/b);
22 }
23 }
```

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>gcc FunEstatica.c Calculadora.c -o exe
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>exe
5 + 7 = 12
6 * 8 = 48
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>
```

```
File Edit Selection Find View Goto Tools Project Preferences Help
FunEstatica.c x Calculadora.c x
1 //##### calculadora.c #####
2 #include <stdio.h>
3 /*
4  Este programa contiene el método principal, el cual invoca a las funciones
5  del archivo funcEstatica.c.
6  */
7 int suma(int,int);
8 //static int resta(int,int);
9 int producto(int,int);
10 //static int cociente (int,int);
11 int main(){
12     printf("5 + 7 = %i\n",suma(5,7));
13     //printf("9 - 77 = %d\n",resta(9,77));
14     printf("6 * 8 = %i\n",producto(6,8));
15     //printf("7 / 2 = %d\n",cociente(7,2));
16 }
```

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>gcc FunEstatica.c Calculadora.c -o exe
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>exe
5 + 7 = 12
6 * 8 = 48
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\Ejemplos>
```

Cuando se compilan los dos archivos al mismo tiempo (gcc funcEstatica.c calculadora.c -o exe), las funciones suma y producto son accesibles desde el archivo calculadora y, por tanto, se genera el código ejecutable. Si se quitan los comentarios y se intenta compilar los archivos se enviará un error, debido a que las funciones son estáticas y no pueden ser accedidas fuera del archivo funcEstaticas.c.

Conclusiones

En los cursos que anteriormente había tomado, no me habían explicado varias cosas acerca de las funciones, en esta práctica aprendí el uso de ellas, como declarar diferentes tipos de variables, así como el uso de la firma de una función. También se vio como definir elementos estáticos. Considero que lo aprendido en esta práctica, me será de gran ayuda al momento de crear programas.

Referencias

- Manual de prácticas del Laboratorio de Fundamentos de programación. 06 Enero 2021, de Facultad de Ingeniería Sitio web: <http://lcp02.fi-b.unam.mx/>