



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Rodríguez Espino Claudia Ing.

Asignatura: Fundamentos de Programación

Grupo: 1104

No de Práctica(s): 12

Integrante(s): Santa Rosa Ortiz Thelma Jazmín.

*No. de Equipo de
cómputo empleado* 50

Semestre: 1°

Fecha de entrega: 03 de noviembre del 2018

Observaciones:

CALIFICACIÓN: _____

Práctica 12. 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.

Desarrollo:

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.

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 tipo de dato puede ser cualquiera de los vistos hasta el momento (entero, real, carácter o arreglo) y el nombre debe seguir la notación de camello. Los parámetros de una función son opcionales.

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.

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 y variables globales.

Como ya se vio, un programa en C puede contener varias funciones. 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;  
}
```

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 ...

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.

```
Argumentos.cpp
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
12     printf("Los elementos del arreglo argv son:\n");
13     for (int cont = 0 ; cont < argc ; cont++){
14         printf("argv[%d] = %s\n", cont, argv[cont]);
15     }
16
17     return 88;
```

Símbolo del sistema

```
29/10/2018 09:49 a.m. <DIR> .
07/09/2018 09:14 a.m. <DIR> ..
29/10/2018 09:49 a.m. 409 Argumentos.cpp
17/09/2018 09:13 a.m. 160 area.c
17/09/2018 10:02 a.m. 7,178 a.out
07/03/2017 01:35 p.m. <DIR> Escritorio
29/10/2018 09:49 a.m. 48,608 Argumentos.exe
29/10/2018 09:48 a.m. 48,636 Argumentos funcion main.exe
17/09/2018 10:02 a.m. 366 ecuacion.c
17/09/2018 09:39 a.m. 223 suma.c
17/09/2018 10:03 a.m. 7,178 ecuacion.out
29/10/2018 09:48 a.m. 409 Argumentos funcion main.cpp
17/09/2018 09:19 a.m. 7,014 area.out
17/09/2018 09:40 a.m. 7,082 suma.out
      11 archivos 127,263 bytes
       3 dirs 363,422,949,376 bytes libres

Z:\>Argumentos.exe 255,45
Los elementos del arreglo argv son:
argv[0] = Argumentos.exe
argv[1] = 255,45
```

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.





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.

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`.

-----MENU CALCULADORA (con funciones)-----

[*] Menu Cal Funciones.cpp Funcion con funcion.cpp 6.1 Elevar al cubo funciones.cpp 6.2 Elevar al cubo

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<math.h>
4  #include<stdlib.h>
5  void suma();
6  void resta();
7  void division();
8  void multiplicacion();
9  int a,b;
10 main()
11 {
12     do
13     {
14         printf("\t\tMENU CALCULADORA\n");
15         printf("\t1. Suma\n");
16         printf("\t2. Resta\n");
17         printf("\t3. Multiplicacion\n");
18         printf("\t4. Division\n");
19         printf("\n\tDime la operacion que deseas realizar: ");
20         scanf("%d",&a);
21
22         switch(a){
23             case 1:
24                 suma();
25                 break;
26             case 2:
27                 resta();
28                 break;
29             case 3:
30                 multiplicacion();
31                 break;
32             case 4:
33                 division();
34                 break;
35             default:
36                 system ("cls");
37                 printf("\t-Esta opcion no existe-\n");
38         }
39         printf("\n\tQuieres intentarlo de nuevo? SI= 1 NO=0?\n\t");
40         scanf("%d", &b);
41         system ("cls");
42     }
43     while(b==1);
44     if(b==0)
45         printf("\n\n\n\t---Gracias por usar este programa <3---\n\n\n");
46     else
47         printf("\n\n\n\tNo existe esta opcion ;,v\n\n\n");
48 }
49 void suma()
50 {
51     system ("cls");
52     float x,y,z;
53     printf("\n\t\tSUMA\n");
54     printf("\tDame primera cantidad: ");
```

 Compile Log  Debug  Find Results  Close

E:\Programacion\Menu Cal Funciones.exe

MENU CALCULADORA

1. Suma
2. Resta
3. Multiplicacion
4. Division

Dime la operacion que deseas realizar: 1

E:\Programacion\Menu Cal Funciones.exe

SUMA

Dame primera cantidad: 56

Dame segunda cantidad: 34

El resultado es:90.000000

Quieres intentarlo de nuevo? SI= 1 NO=0?

E:\Programacion\Menu Cal Funciones.exe

RESTA

Dame primera cantidad: 6

Dame segunda cantidad: 93

El resultado es:-87.000000

Quieres intentarlo de nuevo? SI= 1 NO=0?

E:\Programacion\Menu Cal Funciones.exe

MULTIPLICACION

Dame primera cantidad: 6

Dame segunda cantidad: 4

El resultado es: 24.00

Quieres intentarlo de nuevo? SI= 1 NO=0?

E:\Programacion\Menu Cal Funciones.exe

DIVISION

Dame el dividendo: 10

Dame el divisor: 2

El resultado es:5.00

Quieres intentarlo de nuevo? SI= 1 NO=0?

E:\Programacion\Menu Cal Funciones.exe

---Gracias por usar este programa <3---

-----ELEVAR AL CUBO (con funciones)-----

6.1 Elevar al cubo funciones.cpp 6.2 Elevar al cubo parametros.cpp Funcion con fu

```
1  #include<stdio.h>
2  #include<math.h>
3  #include <conio.h>
4
5  void potencia();
6  int y,a;
7  main()
8  {
9      printf("\n\t\tCUBO DE UN NUMERO\n");
10     potencia();
11     getch();
12 }
13
14 void potencia()
15 {
16     .....
17     printf("\tDame el numero a elevar: ");
18     scanf("%d",&y);
19     a=pow(y,3);
20     printf("\tEl numero al cubo es: %d\n",a);
21 }
```

E:\Programacion\6.1 Elevar al cubo funciones.exe

CUBO DE UN NUMERO
Dame el numero a elevar: 6
El numero al cubo es: 216

-----ELEVAR AL CUBO (con parámetros)-----

6.2 Elevar al cubo parametros.cpp Funcion con funcion.cpp

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<math.h>
4  void potencia(int m);
5  int y,a;
6  main()
7  {
8      printf("\n\t\tELEVAR AL CUBO\n");
9      printf("\tDame el numero a elevar: ");
10     scanf("%d", &y);
11     potencia(y);
12     getch();
13 }
14
15 void potencia(int m)
16 {
17     a=pow(m,3);
18     printf("\tEl numero al cubo es : %d",a);
19 }
```

E:\Programacion\6.2 Elevar al cubo parametros.exe

ELEVAR AL CUBO
Dame el numero a elevar: 6
El numero al cubo es : 216

FUNCION LLAMA FUNCION

Funcion con funcion.cpp

```
1  #include<stdio.h>
2  #include<math.h>
3  #include <conio.h>
4
5  void dat();
6  void retorn();
7  int y,x;
8
9  main()
10 {
11     printf("\n\t\tFUNCIONES\n");
12     dat();
13     retorn();
14     y=(x*100);
15     printf("\n\tEl resultado es: %d",y);
16 }
17
18 void retorn()
19 {
20     printf("\n\tEl numero es: %d",x);
21     getch();
22 }
23
24 void dat()
25 {
26     printf("\n\tDame un numero a multiplicar: ");
27     scanf("%d",&x);
28     getch();
29 }
```

E:\Programacion\Funcion con funcion.exe

FUNCIONES

Dame un numero a multiplicar: 6

El numero es: 6

El resultado es: 600

Conclusión:

Gracias a esta practica, logramos elaborar programas en C donde la solución del problema necesitaba ser dividida en funciones. Distinguimos lo que es prototipo o forma de una función y la implementación de ellas, y manipulamos parámetros en la función principal como en otras.