INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Práctica No. 4.Uso de funciones.

Unidad Temática: **III. Manejo de Funciones y cadenas**

Lugar de realización: Laboratorio de Cómputo **Duración:** 9.0 hrs.

Objetivo

Desarrollar los programas hechos en las prácticas anteriores aplicando funciones y desarrollo de aplicaciones utilizando funciones recursivas.

Resultados Esperados

- Trabajo de investigación logrado en el pre-reporte
- Solución de casos prácticos en Lenguaje C haciendo uso de funciones para la simplificación y eficiencia del código.
- Puesta en práctica las ventajas de usar funciones como son: reutilización de código, aislar mejor los problemas, facilitar el mantenimiento del programa entre otras, al solucionar los casos prácticos planteados.

Pre-reporte.

Para el buen desarrollo de la práctica el alumno entregará un trabajo previo que incluya los siguientes conceptos (**máximo 3 cuartillas**):

1. Mostrar un ejemplo de las dos formas de implementar funciones en un programa en C.
2. Investigar los tipos de paso de parámetros: por referencia y por copia.

"La inteligencia es la función que adapta los medios a los fines"
(Nicolai Hartmann)

Material y Equipo.

- IDE Dev-Cpp o compatible.
- Computadora.

Introducción.

Función

Una función definida por el usuario es una sección de código independiente con nombre, que ejecuta una tarea específica y opcionalmente devuelve un valor.

Sus ventajas son:

- Aislar mejor los problemas
- Producir programas más fáciles de mantener
- Reutilización de código.

Estructura de una función:

```
tipo_retorno nombre_funcion(lista_de_parámetros)
{
      cuerpo de la función;
      return expresión;
}
```

- `tipo_de_retorno:` Tipo de valor devuelto por la función o void si la función no devuelve ningún valor.
- `nombre_función:` Identificador o nombre de la función.
- `lista_de_parámetros:` Lista de declaraciones de los parámetros recibidos por la función separados por comas.
- `expresión:` Valor que devuelve la función.

Ejemplo:

```
float suma(float num1, float num2){
      float resp;
      resp = num1 + num2;
      return resp;
}
```

Una llamada a una función produce la ejecución de las sentencias del cuerpo de la función y cuando se termina la función o se encuentra un return, regresa el control al punto en que fue llamada.

Los Tipos de retorno, pueden ser:

- Tipos de datos simples (int, float, double, char, etc.)
- Apuntador (tipo de dato que permite declarar variables que guardan la dirección en memoria de la variable a la que apuntan, **este tema se abordará más adelante**)

- Estructura (tipo de dato que permite crear una colección de elementos cada uno de los cuales pueden ser de diferente tipo de dato, **este tema se abordará más adelante**)

Ejemplo:

```
int max(int x, int y); //devuelve un entero
double *media(void);   /*devuelve un puntero a double*/
void MAX(int m, float n); /*no devuelve un valor*/
struct InfoPersona buscar(int n) /*devuelve una estructura*/
```

Desarrollo.

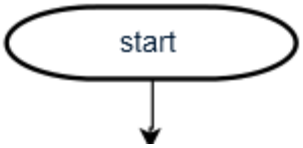
Ejemplo:

Dado un número entero determinar cuántos dígitos tiene. Crear una función para resolverlo.

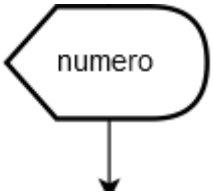
Solución:

Diagrama de Flujo

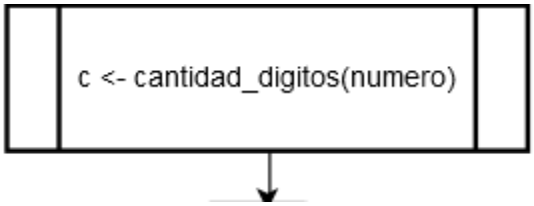
1. Inicio del programa principal



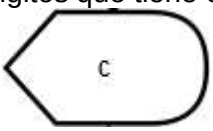
2. Identificación de entradas:
   La entrada ingresada por el usuario es: `numero`



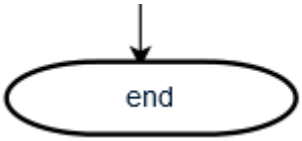
3. Llamar a la función:



4. Identificación de salida
   La salida será el número de dígitos que tiene el número

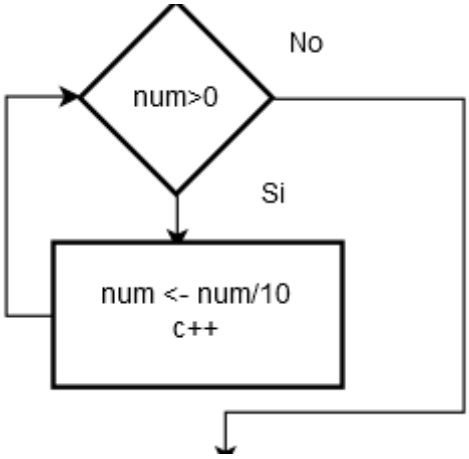


5. Fin

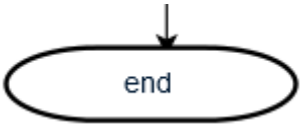


6. Inicio de la función cantidad_digitos(int num)

7. Realizar condición:



8. Retornar `c`
9. Fin



Programa en C *Ver código 4digitos.c anexo en disco*

5. Inicio del programa

```
#include <stdio.h>
```

6. Definición de la función:

```
int cantidad_digitos(int num){
     int c=0;
     while(num>0){
          num = num / 10;
          c++;
     }
     return c;
}
```

7. Identificación de entradas:
   Declarar las variables a utilizar en el programa

```
int main(){
int numero, c;
```

   Solicitar el número al usuario:

```
scanf("%d", &numero);
```

8. Llamar a la función e imprimir el número de dígitos:

```
c = cantidad_digitos(numero);
printf("Cantidad de digitos: %d\n", c);
```

9. Fin

```
}
```

Programas

Realice los siguientes ejercicios:

1. Escribir un programa que calcule el volumen de tres habitaciones diferentes. Asuma que cada habitación es un rectángulo. El usuario introducirá el alto, el ancho y la longitud de cada habitación. El programa mostrará el volumen de cada habitación así como el volumen total de las tres habitaciones.
   Nota: Utilizar una función que calcule el volumen de la habitación.

   Entrada:
   Valor de las variables: `alto, ancho, y longitud.`
   Procedimiento:
   1) **Desarrollar la función calcular_volumen**
   2) **Declarar las variables alto, ancho y longitud.**
   3) **Introducir el alto, ancho y longitud para cada habitación.**
   4) **Calcular el volumen de cada habitación llamando a la función, así como el volumen total.**
   5) **Mostrar los resultados.**

   Salida:
   Suponga los valores de entrada:
   Habitación 1: alto= 2 m, ancho = 20 m, longitud = 10 m
   Habitación 2: alto= 2 m, ancho = 15 m, longitud = 20 m
   Habitación 3: alto= 2 m, ancho = 16 m, longitud = 30 m
   El resultado sería:
   Habitación 1: Volumen = 400 m³
   Habitación 2: Volumen = 600 m³
   Habitación 3: Volumen = 960 m³
   Volumen total: 1960 m³

2. Escribir un programa que, al recibir como dato un número entero N, obtenga el resultado de la siguiente serie:
   Nota: El programa deberá usar una función que calcule el resultado de la serie.

   $$1^1 - 2^2 + 3^3 - \cdots \pm N^N$$

   Entrada:
   Valor de la variable: `n.`
   Procedimiento:
   1) **Desarrollar la función calcular_serie**
   2) **Declarar la variable n.**
   3) **Introducir el valor de la variable n.**
   4) **Calcular el resultado de la serie llamando a la función.**
   5) **Mostrar el resultado.**

   Salida:
   Suponga el valor de entrada:
   n= 5

   El resultado sería:
   Resultado = 2893

3. Escribir un programa que calcule el resultado de elevar un número a una potencia, utilice una función llamada `entero_potencia` la cual reciba como parámetros la base y el exponente y regrese el resultado.

Entrada:

Valor de la variable: `base y exponente.`

Procedimiento:

1) **Desarrollar la función `entero_potencia`**
2) **Declarar las variables base y exponente**
3) **Introducir el valor de las variables base y exponente.**
4) **Calcular el resultado de la potencia llamando a la función.**
5) **Mostrar el resultado.**

Salida:

Suponga el valor de entrada:
base= 2, exponente = 5

El resultado sería:
Resultado = 32

4. Escribir un programa que implemente tres funciones, una para calcular la resistencia, otra para calcular el voltaje y otra para calcular la corriente, la Ley de Ohm está dada por la siguiente formula:

R= V/I

Nota: Mostrar un menú para que el usuario pueda seleccionar lo que desea calcular.

Entrada:

Valor de la variable: `resistencia, voltaje o corriente` y la opción del menú.

Procedimiento:

1) **Desarrollar las funciones, `calcular_resistencia, calcular_voltaje, calcular_corriente.`**
2) **Declarar las variables `resistencia`, `voltaje`, `corriente` y `opcion`.**
3) **Mostrar menú e introducir la opción que se desea calcular.**
4) **Calcular la opción seleccionada llamando a la función.**
5) **Mostrar el resultado.**

Salida:

Suponga que se desea calcular la resistencia y los siguientes valores de entrada:
voltaje = 5, corriente = .3
El resultado sería
16.66 Ohms

Suponga que se desea calcular el voltaje y los siguientes valores de entrada:
resistencia = 50, corriente = 2
El resultado sería
100 Volts

Suponga que se desea calcular la corriente y los siguientes valores de entrada:
voltaje = 10, resistencia = 15
El resultado sería
0.66 Amperes

5. Defina una función llamada `hipotenusa` que calcule la longitud de la hipotenusa de un triángulo recto, cuando se proporcionen las longitudes de los otros lados. Usar esta función en un programa para determinar la longitud de la hipotenusa de

tres triángulos. La función debe recibir dos argumentos `double` y devolver la hipotenusa como `double`.

Entrada:
Valor de la variable: `lado1 y lado2` de cada triángulo.
Procedimiento:
1) **Desarrollar la función hipotenusa**
2) **Declarar las variables `lado1 y lado2`**
3) **Introducir el valor de las variables `lado1 y lado2` para cada triángulo.**
4) **Calcular el resultado de la hipotenusa llamando a la función.**
5) **Mostrar los resultados.**

Salida:
Suponga los valores de entrada:
Triángulo 1: lado1= 3.0 m, lado2 = 4.0 m
Triángulo 2: lado1= 5.0 m, lado2 = 12.0 m
Triángulo 3: lado1= 8.0 m, lado2 = 15.0 m
El resultado sería:
Triángulo 1: Resultado = 5.0 m
Triángulo 2: Resultado = 13.0 m
Triángulo 3: Resultado = 17.0 m

6. Escribir un programa que implemente la función `factorial` para que calcule de manera recursiva el factorial de un número N.

Entrada:
Valor de la variable: `n`.
Procedimiento:
1) **Desarrollar la función factorial recursiva.**
2) **Declarar la variable n.**
3) **Introducir el valor de la variable n.**
4) **Calcular el resultado de factorial de un número llamando a la función.**
5) **Mostrar el resultado.**

Salida:
Suponga el valor de entrada:
n = 5
El resultado sería:
Resultado = 120

Problema de aplicación

Determine las variables de entrada, las variables fijas y las de salida para el análisis del siguiente problema. Además diseñe e implemente en el lenguaje C la solución haciendo uso de al menos 4 funciones:

Sistema de levantamiento sincrónico:

Se requieren válvulas de control para que los cilindro operen de manera uniforme de modo que la carga se levante a la misma velocidad en cada punto. Se cuenta con cuatro cilindros, la operación de éstos puede variar entre 1 y 4 simultáneamente, en caso de ser simultáneo se especifican los parámetros (incluida la carga) para cada uno de ellos. Se requiere obtener la presión hidráulica de trabajo considerando la fuerza:

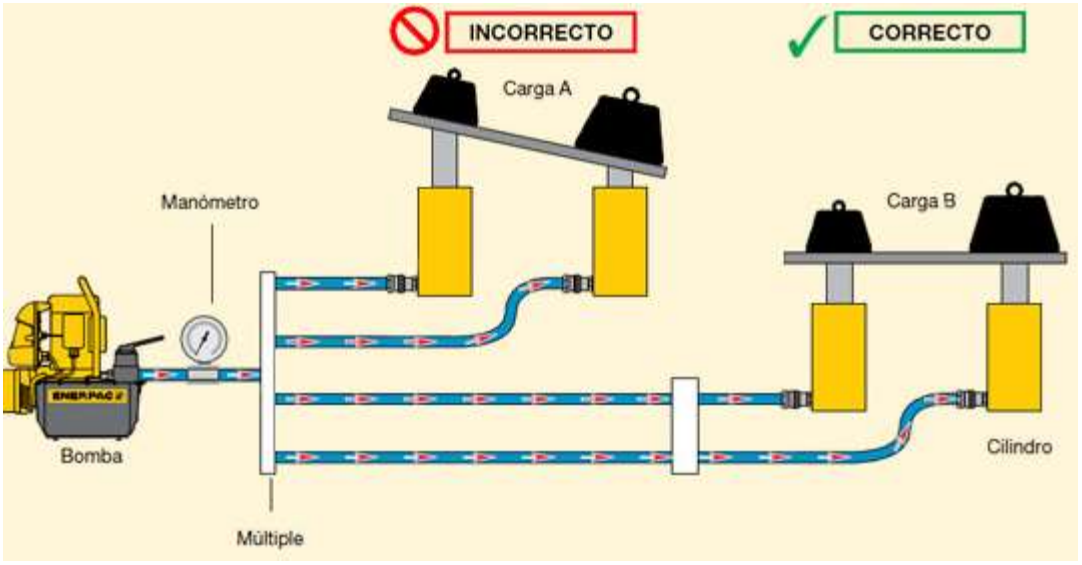


Figura 4.1. Válvulas de control

Fuerza = presión hidráulica de trabajo x área efectiva del cilindro
(lbs (psi) ($inch^2$)

Así como el volumen de aceite del cilindro dado par al fórmula:

Volumen de aceite del cilindro = área efectiva x carrera del cilindro
($inch^3$) ($inch^2$) (inch)

Proyectos (opcionales).

1. Programa que a través de funciones convierta coordenadas polares a rectangulares, es decir, desarrollar una función que calcule el valor de la coordenada en x y otra que calcule el valor de la coordenada en y, dichas funciones deberán de recibir como parámetros el radio (r) y el ángulo θ y calcular las coordenadas en base a las fórmulas mostradas en la imagen:

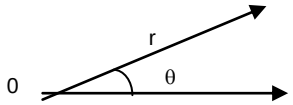


Ilustración 4.2. Ángulo

$$x = r\cos\theta$$
$$y = r\,sen\theta$$

2. Programa que declare un vector de seis elementos y a través de un menú llame a las siguientes funciones:

- `void inicializar_vector(int vector[])`: esta función se encarga de inicializar el vector en cero.
- `void leer_vector(int vector[])`: esta función se encarga de llenar el vector con valores introducidos por el teclado.
- `void imprimir_vector(int vector[])`: esta función se encarga de imprimir el vector.

3. Programa que a través de una función que reciba una cadena de caracteres y la imprima en forma inversa, ejemplo: “hola” se convierte en “aloh”.

Ponderación de la Práctica.

Sección	Elemento a Evaluar
Pre-reporte	Ejemplo de funciones y paso de parámetros
Ejercicio 1	Uso y aplicación de funciones
Ejercicio 2	Recursividad
Ejercicio 3	Uso y aplicación de funciones
Ejercicio 4	Llamado entre funciones
Ejercicio 5	Uso y aplicación de funciones
Ejercicio 6	Recursividad
Problema	Identificación de entradas y salidas, solución de problemas, manejo de funciones

Bibliografía.

Deitel P.J.y Deitel H. M., Como Programar C++, Ed. Prentice Hall, 6 ª Impresión, México, 2009, ISBN: 970-26-1273-X, Págs: 1-1050.

Joyanes Aguilar Luis , Fundamentos de Programación, Ed Mc Graw Hill, 4ª Impresión, España, 2008, ISBN: 8448161114, Págs: 47-73, 74-75, 76-101, 113-141 y 151 -534.

Guardati, Silvia, Estructura de Datos Orientado a Objetos con C ++, Ed.Prentice, México 2007, ISBN: 9702607922, Págs: 1–183.

Cairó Battistuti, Fundamentos de Programación. Piensa en C, Ed. Prentice Hall, México 2006, ISBN: 970-26-0810-4

Antonakos James, Mansfiel Kenneth, Programación Estructurada en C, Ed. Prentice Hall, Madrid 1997, ISBN: 0-13-520487-9

Marcelo Villalobos Ricardo, Fundamentos de programación C++ más de 100 algoritmos codificados, Ed. MACRO, 1ª Impresión, Lima, Perú, 2008, ISBN: 978-603-4007-99-4.

NOTA: Presentar el reporte en un documento (PDF o DOC), el código fuente y la impresión de la pantalla de ejecución.

"Puedes apoyarte de tus compañeros y profesor para aclarar tus dudas"