Fundamentos de Programación Ciclo 02/2021
Laboratorio 2: Programación básica en Lenguaje C

Funciones Principales

- **printf:**

es una función especial porque recibe un número variable de parámetros. El primer parámetro es fijo y es la cadena de formato. En ella se incluye texto a imprimir literalmente y marcas a reemplazar por texto que se obtiene de los parámetros adicionales

```
printf("El valor es %d.\n", contador);
```

- **scanf:**

permite leer varios tipos de datos de una sola vez, tales como enteros, números decimales o cadenas de caracteres

```
scanf("%d", contador);
```

Tipos de Variables

- **Carácter (char)**

un carácter solo sirve para representar una sola letra del alfabeto y este va delimitado por comillas simples
Estas son algunas de las formas válidas para declarar este tipo de datos:

```
char  nombre;
char inicial='H';
char valor,precio,cantidad;
```

En el caso de las funciones básicas tales para leer datos como para mostrarlo se utiliza el carácter especial de “%c”

```
scanf("%c",&nombre);
printf("%c\n",nombre);
```

- **Arreglo ([ ]):**

Un array (unidimensional, también denominado vector) es una variable estructurada formada de un número "n" de variables simples del mismo tipo que son denominadas los componentes o elementos del array. El número de componentes "n" es, entonces, la dimensión del array.
Existen arreglos de cualquier tipo de dato básico, a continuación se muestran algunas de las posibles formas existentes:

```
char saludo[4]="Hola";
int cantidad[3]={1,2,3};
float precios[3]={1.5,2.32,3.453};
```

- **Cadena de caracteres (char[ ]):**

Una cadena de caracteres es un número de caracteres consecutivos (incluso ninguno) encerrado entre unos delimitadores determinados, que en el lenguaje C son las comillas dobles.

Para definir variables de tipo cadena, estas se definen como vectores de caracteres, esto es, anteponiendo la palabra reservada char al identificador de la variable, y después entre corchetes la longitud máxima de cadena.

Estas son algunas de las formas válidas para declarar este tipo de datos:

```
char  nombre[10];
char saludo[4]="Hola";
char valor[15],precio[16],cantidad[20];
```

En el caso de las funciones básicas tales para leer datos como para mostrarlo se utiliza el carácter especial de “%s”

```
scanf("%s",&nombre);
printf("%s\n",nombre);
```

- **Entero (Int):**

representa números enteros con o sin signo, que estarán compuestos por los dígitos del 0 al 9, pudiendo ser precedidos por los signos + o -.

Estas son algunas de las formas válidas para declarar este tipo de datos:

```
int costo;
int edad=24;
int valor,precio,cantidad;
```

En el caso de las funciones básicas tales para leer datos como para mostrarlo se utiliza el carácter especial de “%d” o “%i”

```
scanf("%d",&costo);
printf("%d\n",costo);
```

- **Entero Corto (Short)**

La utilidad de este tipo de datos permite cierto ahorro de memoria permitiendo que el programa al ejecutarse realice un menor gasto de memoria ram ,tanto short como int pueden guardar el mismo valor absoluto máximo, sin embargo debido al mejor uso de memoria que realiza este tipo de dato, es útil en ocasiones que los arreglos de números se aumenten considerablemente de tamaño y el ahorro de memoria permitido por este tipo de dato permite no saturar la computadora en el cual se ejecuta

- **Entero Largo (Long)**

Caso contrario al short los datos de tipo long permiten guardar datos de mayor tamaño que los int, sin embargo, esto ocasiona un mayor uso de memoria de ram por tanto en un long se puede guardar un máximo valor de 4,294,967,295 y en un int solo se puede llegar al valor máximo de 65535

- **Boleano(bool)**

Este tipo de datos es usado principalmente con estructuras de control para así permitir variar el camino de ejecución de programa este tipo de datos se define con la palabra reservada “bool” y solamente puede guardar los datos de true y false

- **Número de punto flotante(Float)**

Se emplean para representar números reales (con decimales).
Estas son algunas de las formas válidas para declarar este tipo de datos:

```
float costo;
float edad=12.5;
float valor,precio,cantidad;
```

En el caso de las funciones básicas tales para leer datos como para mostrarlo se utiliza el carácter especial de “%f”

```
scanf("%f",&costo);
printf("%f\n",costo);
```

- **Número de punto flotante de doble precisión (Double)**

  La diferencia entre Double y Float es la cantidad de bits que utilizan por tanto como vimos anteriormente a mayor cantidad de bits, mayor es los valores que se pueden guardar en el caso de estos tipos de datos, una mayor cantidad de bits ayuda a guardar más decimales por tanto presentar resultados mucho más precisos, con respecto al tipo de datos float este guarda 32 bits dentro de los cuales se distribuyen

con 23 bits de significancia/precisión, 8 bits de exponente y 1 bit de signo, en el caso de los double se duplica este número hasta 64 de los cuales 52 son de significancia/precisión, 11 de exponente y 1 bit de signo.

Tipos de Operaciones

- **Suma(+)**

Esta operación lo que permite es realizar operaciones de suma aritmética tanto con números positivos,negativos también incluyendo decimales, la operación se realiza con el operador especial “+”.

A continuación se presenta un ejemplo utilizando este operador especial:

```
int n1, n2,  suma;
 printf( "\n   Introduzca primer numero (entero): " );
 scanf( "%d", &n1 );
 printf( "\n   Introduzca segundo numero (entero): " );
 scanf( "%d", &n2 );
 suma = n1 + n2;
 printf( "\n   La suma es: %d", suma );
```

- **Resta(-)**

Esta operación lo que permite es realizar operaciones de resta aritmética tanto con números positivos,negativos también incluyendo decimales, la operación se realiza con el operador especial “-”.

A continuación se presenta un ejemplo utilizando este operador especial:

```
int n1, n2,  resta;
 printf( "\n   Introduzca primer numero (entero): " );
 scanf( "%d", &n1 );
 printf( "\n   Introduzca segundo numero (entero): " );
 scanf( "%d", &n2 );
 resta = n1 - n2;
 printf( "\n   La resta es: %d", resta );
```

- **Multiplicación(*)**

Esta operación lo que permite es realizar operaciones de multiplicación aritmética tanto con números positivos,negativos también incluyendo decimales, la operación se realiza con el operador especial “*”.

A continuación se presenta un ejemplo utilizando este operador especial:

```
int n1, n2,  multiplicacion;
 printf( "\n   Introduzca primer numero (entero): " );
 scanf( "%d", &n1 );
 printf( "\n   Introduzca segundo numero (entero): " );
 scanf( "%d", &n2 );
 multiplicacion = n1 * n2;
 printf( "\n   La multiplicacion es: %d", multiplicacion );
```

- **División(/)**

Esta operación lo que permite es realizar operaciones de división aritmética tanto con números positivos,negativos también incluyendo decimales, la operación se realiza con el operador especial “/”.

A continuación se presenta un ejemplo utilizando este operador especial:

```
int n1, n2,  division;
 printf( "\n   Introduzca primer numero (entero): " );
 scanf( "%d", &n1 );
 printf( "\n   Introduzca segundo numero (entero): " );
 scanf( "%d", &n2 );
 division = n1 / n2;
 printf( "\n   La division es: %d", division );
```

Nota: recordar que debemos validar que el número que se encuentra en el denominador no sea cero ya que esa es una división sin solución

- **Módulo(%)**

Esta operación lo que permite es realizar operaciones de división aritmética con la diferencia sobre la anterior explicada, esta no muestra el resultado de la división en sí, sino más bien calcula el resto, por ejemplo, en caso se realiza la operación de 12 módulo 10, lo que estaría dando de resultado es el número 2 ya que es la parte que sobra de la división, la operación se realiza con el operador especial "%".

A continuación se presenta un ejemplo utilizando este operador especial:

```
int n1, n2,  division,modulo;
 printf( "\n   Introduzca primer numero (entero): " );
 scanf( "%d", &n1 );
 printf( "\n   Introduzca segundo numero (entero): " );
 scanf( "%d", &n2 );
 division = n1 / n2;
 modulo = n1 % n2;
 printf( "\n   La division es: %d", division );
 printf( "\n   El modulo es: %d", modulo );
```

- **Potenciación(Pow)**

Para realizar este tipo de operación se necesita incluir la librería de cabecera math.h utilizando "#include<math.h>" aquí se encuentra la operación de potenciación así mismo como las operaciones trigonométricas. Se puede investigar más información sobre las diversas funciones de esta librería dentro de Math.h.

Lo que nos permite esta operación es elevar una base a una potencia X, por ejemplo en el caso de 2 elevado al 2 esto sería igual a 4, a continuación se presenta el ejemplo en código

```
#include<iostream>
#include<math.h>
int main(){

   float v, potencia,  resultado;
    printf( "\n   Introduzca el numero base " );
    scanf( "%f", &base );
    printf( "\n   Introduzca el numero al cual desea potenciar " );
    scanf( "%f", &potencia );
    resultado = pow(base,potencia);

    printf( "\n   La potencia es: %f", resultado );


    return 0;
}
```

- **Raíces Cuadradas(sqrt)**

Para realizar este tipo de operación se necesita incluir la librería de cabecera math.h utilizando “#include<math.h>” aquí se encuentra la operación de raíces así mismo como las operaciones trigonométricas. Se puede investigar más información sobre las diversas funciones de esta librería dentro de Math.h.

Lo que nos permite esta operación es sacar la raíz cuadrada del número que se aplica a la funcion como parametro, por ejemplo en el caso de 16 su raíz cuadrada sería igual a 4, a continuación se presenta el ejemplo en código

```
#include<iostream>
#include<math.h>
int main(){

   float base,  resultado;
    printf( "\n   Introduzca el numero a sacar raiz " );
    scanf( "%f", &base );

    resultado = sqrt(base);

    printf( "\n   La raiz cuadrada es: %f", resultado );


    return 0;
}
```

Guía de Ejercicios

1. Realizar un programa que calcule el área de un círculo con radio 5.6 y el volúmen de una esfera con el mismo valor del radio.

2. Crear un programa que calcule el área de un cuadrado con lado=6 cm y el volumen de un cubo con el mismo valor por lado

3. Crear un programa que solicite al usuario ingresar la cantidad de kilómetros recorridos por una motocicleta y la cantidad de litros de combustible que consumió durante ese recorrido. Mostrar el consumo de combustible por kilómetro.

4. Crea un programa que solicite al usuario que ingrese su nombre. El nombre se debe almacenar en una variable llamada nombre. A continuación se debe mostrar en pantalla el texto “Ahora estás en la matrix, [usuario]”, donde “[usuario]” se reemplazará por el nombre que el usuario haya ingresado.

5. Crear un programa que solicite al usuario el ingreso de una temperatura en escala Fahrenheit (debe permitir decimales) y le muestre el equivalente en grados Celsius. La fórmula de conversión que se usa para este cálculo es: Celsius = (5/9) * (Fahrenheit-32)

6. Crear un programa que solicite al usuario dos números y los almacene en dos variables. En otra variable, almacena el resultado de la suma de esos dos números y luego mostrará ese resultado en pantalla. A continuación, el programa debe solicitar al usuario que ingrese un tercer número, el cual se debe almacenar en una nueva variable. Por último, mostrará en pantalla el resultado de la multiplicación de este nuevo número por el resultado de la suma anterior.

7. Una juguetería tiene mucho éxito en dos de sus productos: payasos y muñecas. Suele hacer venta por correo y la empresa de logística les cobra por peso de cada paquete así que deben calcular el peso de los payasos y muñecas que saldrán en cada paquete a demanda. Cada payaso pesa 112 g y cada muñeca 75 g. Escribir un programa que lea el número de payasos y muñecas vendidos en el último pedido y calcule el peso total del paquete que será enviado.