Nombre de la práctica	PROGRAMAS EN C			No.	1
Asignatura:	METODOS NUMERICOS	Carrera:	INGENIERIA EN SISTEMAS COMPUTACIONALES	Duración de la práctica (Hrs)	6

I. **Competencia(s) específica(s):**
Realizar básico en lenguaje de programación C.
II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):
Aula
III. Material empleado:
Laptop
Editor de texto de Language de programación en C.
IV. Desarrollo de la práctica:
El propósito de esta practica consiste en resolver problemas utilizando el lenguaje de programación C. Problemas basicos que se verán continuación:

```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    printf("este es un texto\n");
    printf("que fue impreso en c");
    printf("\nya soy todo un programador");
    printf("\n");
    return 0;
    }
```

Primero debes incluir las bibliotecas de C
Las cuales son: **# include <stdio.h> y # include <stdlib.h>** así mismo implementamos la función Main dentro de **# include <stdio.h>** encontramos el printf que nos sirve para imprimir texto en consola. Para llegar a la final de la función principal **main** se coloca un return que nos va regresar el valor que esta dentro de la función principal.

Como resultado del programa anterior nos debe mostrar esto:

```
este es un texto
que fue impreso en c
ya soy todo un programador
```

```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    // quiero imprimir el numero 3
    printf("%d \n",3);
    // quiero imprimir la suma de 3 mas 4
    printf("%d \n", 3+4);

    printf("la suma de %d + %d es = %d\n", 3,4,3+
      4);
    return 0;
    }
```

En c existen distintos tipos de comentarios para una sola linea o para dos o mas líneas.
Tal es el caso el comentario para una sola linea usando los \\ seguido del comentario

Asi como se menciona en los comentarios este programa realiza muestra el numero 3 ademas de que realiza la suma de 3+4=7

Y para poder hacer esto hacemos uso del especificador %d que básicamente al utilizarlo con printf nos imprime un numero (variables) de tipo entero limitando los decimales.
No es lo mismo usar %d a usar %f o de algún otro tipo ya que cada uno se utiliza para mostrar cantidades con decimal de numero enteros como lo es el caso.

El resultado de este programa es el siguiente:

```
3
7
la suma de 3 + 4 es = 7
```

```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    // quiero imprimir la suma de 5 mas 8
    printf("%d \n", 5+8);

    printf("%d\n", 78787,3259, 78787+3259);
    return 0;
    }
```

Asi como en el programa anterior debemos imprimir ahora la suma de 5+8 y de 78787+3259
Nuevamente hacemos uso del signo %d para obtener datos de tipo entero.

Y ele resultado de esta operacion se muestra aquí:

```
13
78787
```

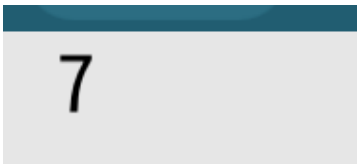
```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    int unNumero = 3;
    int otroNumero =4;

    // quiero imprimir la suma de 3 mas 4
    printf("%d", unNumero + otroNumero);

    printf("\n");
    return 0;
    }
```

En este programa implementamos variables para almacenas datos, las variables en c son un espacio en memoria ram y los nombres de las variables se forman como una secuencia de **letras, números y guiones bajos,** no deben comenzar con un número. ejemplo se puede notar al declarar la variable int unNumero = 3; con tipo de dato entero

El resultado de este programa es el siguiente:




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

int main(){
    int var1 =1;
    int var2 =2;
    int var3 =3;

    printf("%d", var1*var2*var3);
    return 0;
    }
```

Asi como en el programa anterior se declararon 3 variables que almacenen un valor numerico se pretende que se realice una multiplicación. Mostrando el resultado correcto. Simplemente debemos multiplicar la primera variable que almacena el numero 1 por la variable 2 que almacena el valor de 2 por la variable 3 que almacena el valor de 3 y esto da como resultado= 6.

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

int main(){
    double var1 =5.5;
    double var2 = 1.3;
    printf("%f", var1/var2);
    return 0;
    }
```

Ahora haremos una división que muestre el resultado de 5.5 entre 1,3 dividiendo la variable 1 entre la variable 2 en los programas anterior estábamos utilizando el signo %d dentro del printf y ahora se utilizara el signo %f que se utiliza para mostrara resultados con punto decimal. De no hacer uso d estos signos correctamente el resultado se mostrara erróneo. Es por eso que el tipo de especificador importa.

Y el resultado de esta división es el siguiente:

```
4.230769
```

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

int main (){
    float pi=3.141592;
    int radio=5;
    printf("area del circulo");
    printf("\n");
    printf("%f", pi*radio*radio);
    return 0;
    }

```

Debemos imprimir el area de un circulo sabiendo que tiene un diámetro de 10 metros utilizando la formula de area=pi*r2 y así como en el ejercicio anterior implementamos el signo %f para que nos muestre el resultado con decimales. Entonces calculamos el area del circulo de la siguiente manera pi que guarda el valor de 3.141592 por radio por radio que guarda el valor de 5 por que es la mitad del diámetro

Y nos da como resultado lo siguiente:

```
area del circulo
78.539803
```

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

int main (){
    int anios=21;
    int segundos = anios* 365*60*60*24;
    printf("%d",segundos);
    return 0;
}

```

Se requiere calcular los segundos que he vivido así entonces lo que se debe hacer es multiplicar la cantidad de años por los días total del año por lo que es equivalente un minuto por lo equivalente a un segundo por las horas total en un día. Los datos por los que estamos multiplicando son datos enteros así que implementamos el signo %d.

Lo que nos muestra como resultado lo siguiente:

```
662256000
```

Segundos

```c
# include <stdio.h>
# include <stdlib.h>
int main (){
    int var =1;
    int con =1;
    printf("%d",++var);
    printf("%d",var);

    printf("%d",--con);
    printf("%d",con);
    return 0;
    }

```

Implementar el uso de los operador de incremento/decremento al momento de realizar operaciones se ejecuta de la siguiente manera Si es un entero, por ejemplo, la expresión **++var** es equivalente a **(var = var + 1)**. Los operadores de incremento y decremento producen el valor de la variable como resultado. Si el operador aparece antes de la variable (**++var**), la operación se ejecuta primero y después se produce el valor resultante. Si el operador aparece despues de la variable (, **var++**), primero se produce el valor actual, y luego se realiza la operación. Mientras que para el decremento se ejecuta de la misma manera pero en su vez de sumar resultado se resta.

El resultado del incremento de esta operacion es la siguiente:

```
2200
```

```c
# include <stdio.h>
# include <stdlib.h>
int main (){
    int var =1;
    int con =1;
    printf("%d",var++);
    printf("%d",var);

    printf("%d",con--);
    printf("%d",con);
    return 0;
    }
```

El resultado del decremento es el siguiente:

```
2210
```

```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    int x = 5;
    int y = 10;
    int z= ++x * y--;

    printf("x:%d\n",x);
    printf("y:%d\n",y);
    printf("z:%d\n",z);
    return 0;
    }
```

En este programa se tiene que encontrar el valor de z sabiendo que el valor de este mismo es equivalente al incremento de x pero también el decremento de y multiplicando ha ambos para poder obtener el valor de z.
El resultado de z es el siguiente:

```
x:6
y:9
z:60
```

```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    int p = 5;
    int q = 1;
    int r = 2;
    int w = 3;
    int x = 9;
    int y = 6;
    int z;

    z = p*r%q+w/x-y;
    printf("z:%d \n",z);
    return 0;
    }
```

Conocer e interpretar a los operadores en el lenguaje C es importante así sabrás que estas haciendo y si el resultado que obtuviste es correcto. Para eso en este ejemplo se emplea algunos operadores basicos en la operacion donde z almacena el valor de 5*2%1+/3/9-6=-6
Los operadores son
Multiplication =*
division =/
Resta =-
Suma =+
Dividir y tomar residuo (modulo) =%

El resultado de z en esta operacion es el siguiente:

```
z:-6
```

```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    float operacion = (1.0/3.0)+(3.0/5.0)+(1.0/30
      .0)/(23.0/30.0);
    printf("%f",operacion);

    return 0;
    }


```

Implementado el uso de operadores basicos de suma y división para obtener el resultado de estas operaciones. Utilizando el signo de %f por que el resultado que se obtiene es en decimal.

el resultado de la siguiente operacion es:

```
0.976812
```

```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    float operacion = 5.0+(2.0/(1.0/(1.0+2.0/(2.0
      -1.0/4.0))));
    printf("%f",operacion);
    return 0;
}

```

El resultado de la siguiente operacion es:

```
9.285714
```

```c
# include   <stdio.h>
# include <stdlib.h>
int main (){
    printf("and\n");
    printf("true && true :%d\n",(1&&1));
    printf("true&&false :%d\n",(1&&0));
    printf("false&&true :%d\n",(0&&1));
    printf("false&&false :%d\n",(0&&0));


    printf("or\n");
    printf("true||true||:%d\n",(1||1));
    printf("true||false :%d\n",(1&&0));
    printf("false||true :%d\n",(0&&1));
    printf("false||false :%d\n",(0&&0));

    printf("XOR\n");
    printf("true^true :%d\n",(1^1));
    printf("true^false :%d\n",(1&&0));
    printf("false^true :%d\n",(0&&1));
    printf("false^false :%d\n",(0&&0));

    return 0;
    }

```

En este programa se hizo uso de operadores lógicos como lo es de **AND, OR** y **XOR** para realizar operaciones de verdadero o falso según la condición que debía de cumplirse y entonces mostraba 1 o 0 La importancia de realizar este programa es ese saber como funcionan los operadores boleanos. Debemos entender que el AND representado =& que los dos casos deben ser verdaderos para cumplir, para el OR cual quiera de los dos casos puede ser verdadera mientras que para el XOR probando que una de las dos debe cumplir se o en su caso las dos.

Los resultados de la tabla de operadores es la siguiente:

```
and
true && true :1
true&&false :0
false&&true :0
false&&false :0
or
true||true||:1
true||false :0
false||true :0
false||false :0
XOR
true^true :0
true^false :0
false^true :0
false^false :0
```

```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    int p = 1;
    int q = 0;
    int r = 0;
    int t = 0;

    printf("%d\n", p&r);
    return 0;
    }
```

Para este programa se debe Imprimir el resultado de p&r sabiendo que p=1 y r=1 también por lo que se obtiene 0 basándose en lo ya realizado anteriormente con los operadores booleanos.

El resultado de la operación es el siguiente:

```
0
```

```c
# include <stdio.h>
# include <stdlib.h>
int main(){
    printf("%d\n",3>5);
    printf("%d\n",3<5);
    printf("%d\n",3==5);
    printf("%d\n",3!=5);
    return 0;
    }
```

Para este programa se utilizan operadores de comparacion para poder mostrar su funcion utilizando la funcion de operadores lógicos. Los operadores son: mayor que >, menor que < , diferente que ! Mientas que == es para comparar

Nos muestra el los siguientes resultados de estas operaciones:

```
0
1
0
1
```

V. Conclusiones: Es el lenguaje de programacion de proposito general asociado al sistema operativo UNIX trata trata con objetos basicos como caracteres, numeros entre otro tambien con bits y direcciones de memoria pero esto no quiere decir que lo hace mejor o peor que cual quiero otro lenguaje de programación esto solo quiere decir que es distinto que se realizo para implementar en otras aéreas con distintos fines. Esto se define por su estructure por que es distinta ha algunos otros lenguajes. Es importante aprender otra forma de programar saber que o donde lo puedes implementar.