Práctica Básica

Está práctica es para aquellos que necesitan comenzar desde 0 y no es de presentación obligatoria, si ya se tienen conocimientos básicos de programación pueden hacer directamente las actividades para entregar

Lo primero es familiarizarse con la estructura básica de un programa en C/C++

#include <iostream> //incluye la librería de entrada/salida donde está cout

using namespace std; //sino se dice otra cosa se supone las funciones estandar (std)

int main(int argc, char *argv[]) { //Esta es la función principal aca comienza el programa

return 0; //termina la función principal informando que todo se realizó bien

}

Primero vamos a practicar **cout** que como habrán notado sirve para imprimir por pantalla, para ello vamos a escribir cout dentro de la función main y no nos olvidemos del punto y coma:

```
#include <iostream>
 //incluye la librería de entrada/salida donde está cout

using namespace std;
 //sino se dice otra cosa se supone las funciones estandar (std)

int main(int argc, char *argv[]) {
 //Esta es la función principal aca comienza el programa

cout<< "Imprimimos por pantalla";

return 0;
 //termina la función principal informando que todo se realizó bien

}
```

Los literales entre comillas se los conoce como strings o cadena de caracteres y representan exactamente el texto que está entre comillas.

Probemos ahora colocando los couts dentro del main:

```
cout<< "Esto es una linea";
cout<< "Esta es la que sigue";
```

Veamos que pasa si:

```
cout<< "Esto es una linea\n";
cout<< "Esta es la que sigue";
```

y si probamos

```
cout<< "Esto es una linea"<<endl;
cout<< "Esta es la que sigue";
```

y esto:

```
cout<< "Esto es una linea";<<endl<< "Esta es la que sigue";
```

Se puede también colocar números:

```
cout<<5;
```

```
cout<<"5";
```

¿Cual es la diferencia entre ambos?

El 5 sin comillas es el número 5, todo número como es es la representación de su valor. Los números valores se puede operar matemáticamente, los números en las cadenas no.

```
cout<<5+5;
```

```
cout<<"5+5"
```

¿Cuando se usa uno o el otro? El primer caso cuando necesitamos el resultado de la suma, el segundo cuando queremos imprimir la expresión de la suma:

```
cout<<"5+5="<<5+5;
```

Caracteres Especiales

'\n' es el caracter de retorno de carro, o sea el ENTER que nos lleva a una nueva línea como lo habrán probado anteriormente. Pero no es el único, pueden ver algunos en el apunte de Introducción al Lenguaje C/C++, también pueden ver algunos de los modificadores como endl.

Con respecto a los caracteres especiales, como se usa \ para indicar que el que sigue es un caracter especial cuando queremos imprimir realmente la \ hay que poner \\, otros caracteres son por ejemplo si queremos imprimir comillas la manera que no confunda el C/C++ entre las comillas que indican el comienzo y el final de una cadena. Para eso se usa la barra, por ejemplo:

```
cout<<"Queremos imprimir entre comillas \"5+5\"";
```

Lo mismo ocurre con los apóstrofes, los apóstrofes se usan para indicar un caracter, por ejemplo:

```
cout<<'a';
```

imprime el caracter **a**

```
cout<<a;
```

imprime el contenido de la variable **a**

```
cout<<'\';
```

imprime el caracter ' (apóstrofe)

Cálculos matemáticos

Los cálculos matemáticos se realizan como todos de izquierda a derecha, siguiendo las reglas de precedencia matemática, que se pueden alterar usando paréntesis, las operaciones básicas son suma(+), resta(-), multiplicación(*), división(/), resto(%).

cout<<5+4*3;

cout<<(5+4)*3;

Prueben las distintas operaciones, sobre todo el resto, el resto es lo que queda de una división entera por lo que solo funciona con enteros, la división en C/C++ se distingue entre enteros y reales (float) la división entre enteros devuelve el cociente de la misma, para saber el resto hay que usar %, la divisiones entre reales devuelve el resultado con decimales, se distinguen cuando ambos operandos son enteros el resultado es un entero cuando uno de ellos es real el resultado es real.

Probar:

14/4 y 14%4 y 14.0/4.0

Agregando Variables

Como ya vimos las variables sirven para almacenar valores se las debe declarar primero antes de usarlas:

int a; //a solo puede contener enteros
float b; //b puede contener reales

la asignación se realiza con el signo igual

a=1; //ahora a contiene 1
b=1.5; // ahora b contiene 1.5 (se usa punto en vez de coma)

También si no sabemos exactamente el valor podemos plantear el cálculo

a=5+4*3;
b=27.0/4.0;

¿Por qué 27.0 y no 27? Porque la división / da diferente resultado si se hace entre enteros que entre reales, una división entre enteros da como resultado un entero sin la parte decimal, un 27 es un entero 27.0 le indicamos que es un real.

Se puede usar otra variable para asignar:

a=27;
b=a/2.0; //si uno de los elementos de la división es real la división devuelve un real.

Se pueden saber los resultados con cout

```
cout<<a<<endl<<b;
```

Funciones matemáticas

No solo existe la suma (+), resta (-), multiplicación (*) y división (/), sino que hay multiplicidad de funciones matemáticas como las trigonometricas:

sin() y cos()

Nota: para que las funciones matemáticas estén disponibles se debe incluir la librería matemática:

#include<math.h>

en donde van los includes

Las funciones tienen un nombre de la misma, por ejemplo sin para el seno, y un argumento que es uno o más valores que se le pasan a la función para que realice el cálculo, en nuestro caso del seno necesita saber el ángulo del cual queremos calcularlo.

```
sin(5);
```

Las funciones siempre devuelven un valor, en el caso de sin devuelve el seno del argumento, ese valor lo podemos tanto imprimir por pantalla con cout, o asignarlo a una variable float o que sea parte de otro cálculo por ejemplo:

```
cout<<sin(5);
b=sin(5);
cout<<2*sin(5);
b=2*sin(5);
```

También pueden recibir como parámetro otra variable

```
a=5;
b=sin(a);
cout<<b;
```

Nota: las funciones trigonométricas en la mayoría de los lenguajes de programación (incluido el C/C++) no usan grados para los ángulos sino que estos se miden en radianes, para pasar de grados a radianes hay que multiplicar los grados por PI y dividirlos por 180, para pasar de radianes a grados hay que multiplicar por 180 y dividir por PI

```
a=45; //vamos a calcular el seno de 45 grados
b=sin(a*3.1416/180); //antes de calcular el seno convertimos los grados en radianes.
cout<<b; //comparar con una calculadora el resultado
```

prueben cambiar a para ver distintos resultados
prueben cambiar sin por cos y comparar con la calculadora

Otras funciones matemáticas:

sqrt(valor); //devuelve la raíz cuadrada de valor
pow(base,exponente); //devuelve el calculo potencia de base elevado al exponente, 3 al cuadrado sería pow(3,2);

Prueben calculando varios números.
La definición de raíz de una potencia es la misma que la base elevada a uno sobre la raíz, por ejemplo la raíz cubica de 4 es lo mismo que la potencia de 4 elevado a la 1/3

Pueden probar esto haciendo por ejemplo:

sqrt(5)
pow(5,1.0/2.0)

Prueben con varios números

Otras funciones están disponibles, una muy útil es un generador de números al azar, esta función se llama **rand** (de random en ingles que significa aleatorio), no necesita ningún argumento, y cada vez que se llama devuelve un número distinto al azar. Por más que no reciba argumento los paréntesis deben estar para que C/C++ distinga **rand** como una función y no como una variable:

cout<<rand()

Pueden probar llamarla varias veces:

cout<<rand()<<endl<<rand()>>endl<<rand();

Obtendrán tres números al azar.

Pero existe un problema, los números al azar no son fáciles de generar por una computadora, es por eso que se realiza un truco matemático llamado generador de números pseudoaleatorios ver:

http://es.wikipedia.org/wiki/N%C3%BAmero_pseudoaleatorio

http://es.wikipedia.org/wiki/Generador_de_n%C3%BAmeros_pseudoaleatorios

Estos números son muy importantes en los videojuegos por lo que nos viene mal familiarizarse con alguno de sus problemas.

Los números pseudoaleatorios son una serie de números aparentemente al azar, pero si la función matemática que los produce comienza siempre en el mismo número la secuencia es siempre la misma. Esto se puede observar cada vez que se ejecute el programa.

cout<<rand()<<endl<<rand()>>endl<<rand();

Para cambiar el inicio de la secuencia está la función **srand** que tiene como parámetro el valor con el cual se empieza a generar la serie pesudoaleatoria. Claro que si uno coloca siempre el mismo valor siempre obtendrá la misma serie. Para eso se acostumbra a colocar un valor que se sepa cambie siempre cada vez que se ejecute el programa, los más fácil es colocarle la hora del sistema:

srand(time(NULL));

La función **time** devuelve la cantidad de segundos pasados, o sea, cada vez que se llama su valor va cambiando subiendo cada vez que pasa el tiempo.

cout<<time(NULL); //cada vez que se ejecute el programa va a tener un valor diferente de acuerdo al tiempo que va pasando.

Ingresar valores desde el teclado

Para ingresar valores desde el teclado se puede usar **cin**, cin es el "opuesto" de cout, mientras que cout es la salida por pantalla, cin es la entrada por teclado:

```
cin>>a; //espera un entero por teclado
cin>>b; //espera un float por teclado
cout<<"a vale:"<<a<<endl;
cout<<"b vale:"<<b<<endl;
```

cin sabe que valor agregar fijándose en el tipo conque fue declarada la variable que si mal no recuerdan era int a; float b;

Ejercicios de práctica

Hagan un programa que ingrese los grados de un ángulo y los convierta a radianes.
Lo mismo pero que ingrese radianes y muestre grados.
Un programa que ingrese grados centígrados y muestre cuanto es en grados Kelvin y grados Fahrenheit.
Un programa que ingrese los lados de un rectángulo y calcule el perímetro y la superficie. Lo mismo con un triángulo.
Que ingrese el radio de una circunferencia y calcule el perímetro y su superficie, y el volumen de una esfera.
Que calcule el tiempo que tarda un objeto en caer desde una altura dada en el vacío.