Empezando a programar desde cero

Para los que empiezan el curso con 0 de conocimientos de programación, tienen un montón de conocimientos nuevos para asimilar. Por ese motivo vamos a asentar algunos de ellos.

En primer lugar, esta unidad no entra de lleno en lo que es la programación propiamente dicha, esto se verá a partir de la siguiente unidad, lo que haremos aquí es sentar las bases de conceptos para luego más adelante poder manejarnos con la programación propiamente dicha. De todas manera esto no significa que no trabajemos con programas simples.

El primer paso es reconocer como es una estructura de un programa en C/C++, viendo el apunte y comparándolo con el fuente número 1 pueden hacerse una idea de las diferentes partes.

Sin embargo no necesitan estar todas presentes para hacer un programa simple, si han instalado el ZinjaI y accedieron a escribir un programa simple (ver Manual de Instalación del IDE ZinjaI) estarán viendo un programa base como este:

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[]) {

	return 0;
}
```

Lo primero que deben saber es como colocar comentarios, los comentarios son textos dentro del programa que el compilador ignora de tal manera que podemos escribir lo que se nos ocurra, tiene dos usos muy importantes, el primero es dejar aclaraciones para que otro programador pueda entender en forma más cómoda lo que hace el programa, la otra nos da una manera rápida de suprimir instrucciones cuando estamos probando el funcionamiento de nuestro programa sin tener que borrarlas.

Los comentarios se escriben entre **/*** aquí va el comentario ***/** (entre barra asterisco y asterisco barra todo lo que se coloque será ignorado por el compilador), y la otra manera es colocando una doble barra **//** que ignora todo lo que está escrito hasta el final de la línea.

```
#include <iostream>
using namespace std;

/* Este es un ejemplo de escritura de comentarios
todo lo que yo escriba entre barra asterisco y asterisco barra
es ignorado por el compilador */

//todo lo que yo coloque después de las dos barras hasta el final de la línea también es ignorado

int main(int argc, char *argv[]) {

	return 0;
}
```

Si compilamos y ejecutamos este programa usando F9 o el triángulito verde de Play veremos que no marca ningún mensaje de error, aunque nuestro programa no hace nada.

Veamos las partes del mismo

```
#include <iostream> //esta instrucción carga una biblioteca de funciones (lo veremos más adelante)
using namespace std; //indica que usamos el espacio de nombres estándar

int main(int argc, char *argv[]) { //este es el comienzo de la función main
// todo programa de C comienza en la función main

        return 0; //esta instrucción indica que la función termine y devuelva un cero
}

/* como ven nuestro programa empieza en main y de inmediato termina con el return
   lo que está entre paréntesis después de main son los argumentos o valores que recibe la función
   esto se verá en la unidad que trata sobre funciones*/
```

Lo más simple de hacer es escribir en la pantalla, para ello tenemos la instrucción cout:

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[]) {

        cout<<"Escribimos en la pantalla";

        return 0;
}
```

Fijense el detalle de que cada instrucción termina con un **punto y coma**, eso le dice al compilador donde termina cada una de ellas.

Las llaves indican donde empieza y donde termina la función.

El siguiente paso es aprender sobre las variables. Una variable es un lugar en la memoria donde guardaremos un valor para usarlo cuando queramos, además a ese valor podemos modificarlo durante el programa.

Para crear una variable primero debemos saber que tipo de valor vamos a guardar, en C hay algunos tipos básicos, entre ellos:

el entero (**int**) que almacena un valor entero
el caracter (**char**) que almacena un caracter
el real o flotante (**float**) que almacena números reales
el real doble (**double**) que almacena números reales con el doble de precisión que **float**
el booleano (**bool**) que almacena verdadero (**true**) o falso (**false**)

Dependiendo que vamos a almacenar es como vamos a declarar nuestra variable, por ejemplo, solo vamos a trabajar con enteros entonces podemos declarar una variable de nombre a así:

```
int a;
```

si hubiera querido que sea un real la hubiera declarado:

float a;

Para darle valor a nuestra variable se usa el signo igual, por ejemplo si quisiera darle el valor 1 podría hacer:

int a;

a=1;

Si escribimos un programa podría ser:

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[]) {

	int a;

	a=1;

	cout<<"La variable a vale:"<<a;

	return 0;
}
```

Y de paso vemos que **cout** también puede mostrar variables.

No solo podemos asignar números, sino también resultados de cálculos, por ejemplo:

float b;

b=15.0/100;

Ahora la variable **b** contendrá el valor 0,15 o lo que es lo mismo el resultado de dividir 15 por 100. Una característica de C es que la división para **float** es diferente que para la de enteros (se explica en la unidad siguiente) por ese motivo es necesario colocar el 15.0 en vez de 15 solo para que lo diferencie de un número entero.

Por supuesto también podemos imprimirlo en pantalla con cout (prueben siguiendo el ejemplo de la variable **a**)

Las operaciones matemáticas básicas suma (+), resta (-), multiplicación (*) y división (/) se pueden hacer fácilmente, prueben a cambiar la cuenta de b por el cálculo que deseen, solo tengan en cuenta que el símbolo de multiplicación es el asterisco (*) y el de la división es la barra (/), los de suma y resta siguen siendo los mismos.

La otra forma de asignarle un valor a una variable es a través del teclado mediante la instrucción **cin** que funciona parecido a **cout** solo que en vez de imprimir en pantalla toma un valor desde el teclado y se lo asigna a una variable (presten atención los signos de mayor que)

cin>>b;

Para verlo en acción no hay nada mejor que programar:

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[]) {

	int b;

	cout<<”Introduce un numero entero:”; //para tu sepas que hay que hacer

	cin>>b; //espera un número entero por teclado

	cout<<”La variable b vale:”<<b;

	return 0;
}
```

Una variable puede cambiar de valor a medida que pasa el programa, por ese motivo se llama variable. Veamos el siguiente ejemplo:

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[]) {

	float b;

	b=2.5; //para los decimales se usa punto y no coma

	cout<<”La variable b vale:”<<b<<endl; //endl provoca un cambio de linea

	b=3.33;

	cout<<”La variable b vale:”<<b<<endl;

	return 0;
}
```

Una variable puede ser asignada por otra variable:

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[]) {

	int a;
```

```
	float b;

	a=5;
	b=a/2.0;

	cout<<”La variable b vale:”<<b;

	return 0;
}
```

Vemos también en este último ejemplo que una variable puede ser usada como parte de otro cálculo.

Por último, la misma variable puede ser usada para recalcularse a si misma:

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[]) {

	int a;
	float b;

	a=5;
	a=a*2;
	b=a/2.0;

	cout<<”La variable a vale:”<<a<<endl;

	cout<<”La variable b vale:”<<b<<endl;

	return 0;
}
```

Las constantes son similares a las variables solo que cuando se le asigna el valor ya no puede cambiarse más durante el programa.

Estos son los puntos básicos de está unidad, te recomiendo que releas el material de lectura, veas los fuentes y te familiarices con estos cuatro elementos: **cout**, **cin**, **variables** y **constantes**.

Una vez que hayas visto de nuevo todos estos elementos, ya podemos ir a las actividades de esta Unidad.

Como resolver las Actividades

Las actividades de esta unidad, no buscan que crees un programa, sino con los conocimientos que adquiriste modifiques uno que ya está escrito, más precisamente el fuente número 1.

Para poder realizar el primero van a tener que ver un poco las opciones de cout para dar formato a lo que se imprime por pantalla.

El segundo elemento es ver como modificar el cálculo del disparo para que informe de manera diferente (positivo o negativo según quede corto o se pase).

El tercero deberán cambiar una constante por una variable y además usar cin.

En el cuarto, tienen dos desafíos, uno es hacer que la distancia donde se encuentra el blanco se halle entre 1000 y 1500 metros y no como en el fuente original que está entre 100 y 300 metros (tendrán que averiguar como funciona rand). El otro desafío es lograr que la bala llegue a esa distancia, ya que si dejan todo como está seguirá disparando entre 0 y algo más de 300 metros, es un pequeño cambio pero tienen que encontrarlo.

El quinto ya es más fácil ya que es a gusto, se supone que aprovechen lo que aprendieron de lo anterior, y si quieren agregar algo más mejor, solo que no se salgan de cin, cout y variables, nada de bucles for y while ni if o switch, eso se verá en la siguiente unidad.

El sexto puede ser algo complicado, es exactamente el mismo cuerpo del programa pero cambia la perspectiva la nueva fórmula para la caída de la bomba es:

V0*sqrt(2*H/G) donde V0 es la velocidad del avión en m/s, H es la altura de vuelo en m y G es la aceleración de la gravedad en m/s. El resultado de este cálculo es la distancia que recorre la bomba desde que sale del avión hasta que toca el suelo en metros en sentido horizontal.

Es algo parecido al segundo pero con otra fórmula.