UNIVERSIDAD NACIONAL DEL LITORAL
FACULTAD DE INGENIERÍA Y CIENCIAS HÍDRICAS

Tecnicatura en diseño
y programación de Videojuegos

UNLVIRTUAL

Introducción a la programación

Unidad Temática Número 2

Introducción al Lenguaje ANSI/ISO C++

Objetivo: Acercar al Alumno a los Conceptos de C++

Temas: Compiladores e Intérpretes, Código Fuente, Código Objeto, Enlazador, IDE, RAD, Estructura de un Programa C++, Elementos de un Programa C++, Indentificadores, Literales (constantes), Tipos de Datos Estándar, Notación y Definición de Constantes, Declaración e Inicialización de Variable, Entrada y Salida.

INTRODUCCIÓN

En esta Unidad nos introduciremos al lenguaje de programación. Un lenguaje de programación es, precisamente, un lenguaje que nos permite escribir instrucciones de manera tal que, al ser procesadas, podemos controlar los componentes físicos de la computadora, como el procesador, la memoria, la pantalla, el teclado, etc. Así crear programas de software como, en este caso, videojuegos. Existen muchísimos lenguajes de programación, cada uno tiene sus bondades y desventajas. Nosotros utilizaremos C++.
Antes entrar de lleno en el lenguaje vamos a introducir un par de herramientas y conceptos. Lo que nosotros escribimos en el lenguaje de programación se denomina **código fuente**. Podemos escribir código fuente en cualquier procesador de texto como Notepad, Gedit, Word, etc, cualquier programa que nos permita escribir texto y guardarlo nos podría servir. Sin embargo utilizamos un tipo de programa especial denominado **IDE** (Entorno de Desarrollo Integrado, en inglés Integrated Developed Enviroment). El motivo de usar un IDE y no un simple editor de textos es que el IDE ya viene configurado con las herramientas que necesitamos y tiene muchísimas facilidades que nos ayudan infinitamente. Al igual que las primitivas, los lenguajes de programación también tienen instrucciones básicas, claro que en este caso son palabras, las palabras que constituyen primitivas se llaman **palabras reservadas** (ya que no pueden ser utilizadas nada más que para su función). El conjunto de palabras reservadas y reglas para escribir un lenguaje se denomina **sintaxis** del lenguaje. Ya veremos todos estos conceptos a continuación y de manera práctica.

SINTAXIS DE C++

A partir de aquí se sugiere que lean el apunte con un IDE abierto para que puedan probar todas las funcionalidades que aquí se mencionan. En caso de no poder instalar ninguno o no tener ninguno disponible en la PC que se utiliza, se pueden utilizar algunas consolar online como http://cpp.sh/ o https://www.onlinegdb.com/online_c++_compiler. Desde la cátedra se sugiere la instalación de Zinjai http://zinjai.sourceforge.net/. En el aula hay links y videos que explican su funcionamiento.

Todo archivo ejecutable de C++ arranca con la función **main** (principal, en inglés):

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

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

    return 0;
}
```

De todo este palabrerío lo importante es la que está entre llaves **{.}**. Todo lo que nosotros escribamos debe encontrarse ahí adentro (de momento). Vamos a obviar la línea que dice **return 0;** y vamos a escribir encima de ella. Otro concepto, cada "oración" que nosotros escribamos en un lenguaje de programación se denomina **línea**, es muy probable que el IDE que utilicemos tenga las líneas numeradas, esto nos ayuda a guiarnos.
Vamos a escribir todo lo que vayamos aprendiendo en este documento exactamente donde se encuentra la flecha:

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

int main (int argc, char *argv[]) {
➡
    return 0;
}
```

La primera instrucción que vamos a aprender en C++ es la salida en pantalla. Para escribir algo en la pantalla utilizaremos la instrucción **cout** (es importante que para poder utilizar esta función se encuentre la línea **#include<iostream>**). La instrucción cout se utiliza de la siguiente manera:

cout (la instrucción)
<< (menor menor que es el operador de entrada, ya veremos esto más adelante)
"palabra" (la frase o variable que queramos mostrar)

Las palabras que escribimos debe hacerse entre comillas dobles " " así el programa entiende que es una palabra y no otra cosa.
Entonces, para vamos a escribir nuestro nombre:
Importante, toda las líneas en C++ deben finalizar con **punto y coma** (;). Esto es parte de las reglas de sintaxis de C++.

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

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

    cout << "Alejandro";

    return 0;
}
```

Obviamente que cada uno escribirá su nombre. Salvo que se llamen Alejandro. :P

En Zinjai ejecutamos nuestro código con el botón de play: 

Si todo salió bien se abrirá una ventana de comandos con nuestro nombre y quizás alguna leyenda más. En Zinjai la consola diría lo siguiente:

```
Alejandro

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>_
```

Ya escribimos nuestro primer programa en C++. De la misma forma podemos escribir cualquier oración, frase o número.

VARIABLES

Escribir frases es el comienzo, pero todavía hay mucho más por ver.
Las variables constituyen el elemento básico fundamental de cualquier lenguaje o dispositivo. Cuando ponemos una alarma en nuestro celular, cuando llenamos un formulario de registro en alguna página, cuando configuramos el microondas, en todo momento estamos fijándole valores a las variables. La definición es análoga al concepto matemático. Básicamente una variable es un identificador reconocido por el sistema (dispositivo, función, etc) que puede contener un valor que puede o no variar a lo largo de su vida (ámbito).
Un ejemplo práctico del uso de una variable puede ser cuando tomamos un colectivo para llegar a otro lugar. Yo sé que el colectivo 24 me lleva de mi casa a la casa de un amigo, sé que lo tengo que tomar en la esquina de mi casa y me deja en la esquina de la casa de mi amigo. A mí sólo me alcanza con saber que el colectivo es el 24, luego, si es un coche de una marca u otro no me interesa, ni tampoco quien es el conductor, las personas con las que viejo, incluso tampoco me interesa si cambia de recorrido, mientras yo me lo tome en la esquina de mi casa y me deje en la esquina de la casa de mi amigo. En este ejemplo, nosotros seríamos el sistema que usa la variable, y ésta es configurada por todo el resto del contexto, la línea de colectivos, las personas que se suben, el tránsito, etc. Es posible que más de uno se pregunte ¿En qué varía entonces la variable si yo siempre me tomo la misma línea? Bueno, puede ser que dos días seguidos haya tomado el mismo colectivo, manejado por la misma persona, con los mismos pasajeros a la misma velocidad y todo idéntico, podríamos decir que en esos dos viajes el valor (que vendría a ser el viaje) no varió. Pero si llegara a cambiar, en ese caso sí varió. Aunque para nosotros el 24 sigue cumpliendo su función, que es llevarme de mi casa a la casa de mi amigo. Veamos el ejemplo donde nosotros cambiamos el valor de la variable. Siguiendo con los ejemplos de colectivos. Vamos a suponer que voy a viajar con el mismo amigo del ejemplo anterior y vamos a comprar pasajes, nos tocan los asientos 11 y 12. Si fuera otra persona a comprar boletos el sistema de venta diriía que los asientos 11 y 12 están ocupados por lo que no pueden ser vendidos. No importa

quién los ocupan, ni si realmente los van a usar, si son dos personas o si es sólo una que quiso comprar los dos boletos, lo que importa es que están ocupados. Ahí el sistema de la boletaría es claramente el sistema que consulta la variable, y mi amigo y yo somos 2 valores en la vida de la variable. Para el siguiente viaje el asiento 11 y 12 van a estar ocupados por otras personas o vacíos, y van a tener muchos valores a lo largo de la vida útil de ese colectivo.
Volviendo a C++, en C++ hay varios tipos de variables dependiendo el contenido que va a tener. Los tipos de variables son:

Definición tipo	Valores posibles	Tamaño en bytes
Caracteres		
char	256 caracteres del mapa ASCII	1
Números enteros		
short	-32768 : 32767	2
unsigned short	0 : 65535	2
int	- 32768 : 32768	4
unsigned int	0 : 65535	4
long	-2.147.483.648 : 2.147.483.647	8
unsigned long	0 : 4.294.967.295	8
Números reales (con punto flotante)		
float	3.4 x 10-38 : 3.4 x 1038	4
double	1.7 x 10 -308 : 1.7 x 10 308	8
long double	3.4 x 10 -4932 : 3.4 x 10 4932	10
Lógico		
bool	True (cierto) y false (falso)	1
Nulo		
void	---	0

Sobre la tabla. En la primer columna está el tipo de dato primitivo, es decir, la definición del tipo de variable que debemos utilizar para almacenar un valor (ya veremos ejemplos). En la segunda columna los valores que este tipo de dato puede almacenar, en caso de los intervalos, están separados por los dos puntos (:). Por ejemplo, un short puede almacenar valores desde -32768 hasta 32767. En la tercer columna está el tamaño de byte que ocupa ese dato, cuánto más valor necesitemos, más tamaño ocupará en la memoria y más tamaño tendrá nuestro programa final, aun así, el tamaño de la memoria no es algo en lo que nos ocupemos por el momento, incluso en esta materia no nos interesa que los programas sean eficientes, preferimos ocuparnos de la programación. Igualmente el dato del espacio en bytes es importante para tener noción de la diferencia entre un tipo y otro, pero no le daremos mayor importancia de aquí en adelante. El valor del char corresponde al código de la tabla ASCII (https://elcodigoascii.com.ar/). Así el 65 es la A, el 122 la z, etc. En consola, el valor del bool se verá como 1 para true (cierto) y 0 para false (falso).

La sintaxis para declarar una variable es la siguiente:

Tipo de variable, nombre de variable, punto y coma

```
int manzanas;
```

Así tenemos una variable de tipo entero (int) llamada *manzanas*.

El proceso de crear un dato y especificarte un tipo se llama **declaración**. También es posible asignarle un valor en el mismo momento que se crea la variable.

```
int manzanas = 10;
```

Así tenemos una variable de tipo entero (int) llamada *manzanas* y le asignamos el valor de 10.

Existen algunas reglas para los nombres de las variables:

+ Utilizar como primer caracter una letra
+ Continuar con letras, dígitos o guion bajo (_)

+ No utilizar palabras reservadas de C++ (Ver tabla al final de la Unidad)
+ C++ considera diferentes las mayúsculas de las minúsculas
+ No se pueden usar espacios en blanco
+ No se admiten tildes o ñ

En el siguiente ejemplo declararemos y le daremos valor a varias variables en C++, de distintos tipo y las mostraremos en pantalla.

```
#include<iostream>

using namespace std;

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

    char letra = 'a';
    cout << "La primer letra del abecedario es: "<<letra;
    cout<<endl;

    int manzanas = 10000;
    cout << "El tren carga : "<< manzanas << " manzanas";
    cout<<endl;

    short monedas = 510;
    cout << "Monedas juntadas en el nivel: " << monedas;
    cout<<endl;

    bool vida = false;
    cout << "El heroe se encuentra con vida?: " << vida;
    cout<<endl;

    return 0;
}
```

La salida de la consola sería:

```
La primer letra del abecedario es: a
El tren carga : 10000 manzanas
Monedas juntadas en el nivel: 510
El heroe se encuentra con vida?: 0


<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

Algunas aclaraciones sobre el ejemplo:
+ El comando **endl** (end line) se utiliza para agregar un salto de línea
+ Un caracter es una única letra y se coloca entre comillas simples ‘ ‘, en cambio una cadena (un carácter o más de uno) se coloca entre comillas dobles “ “. El char guarda sólo un carácter por lo que siempre debe ir entre comillas simples.

A continuación veremos un ejemplo dónde modificamos el valor variable.

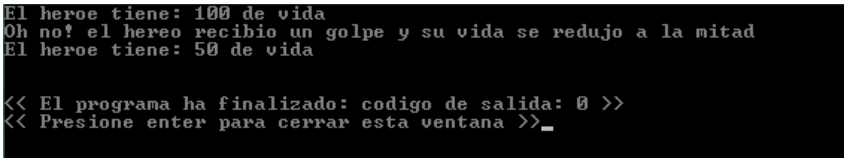
```
#include<iostream>
using namespace std;

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

    int vida = 100;
    cout << "El heroe tiene: "<< vida << " de vida"<<endl;;
    cout << "Oh no! el hereo recibio un golpe y su vida se redujo a la mitad"<<endl;
    vida = 50;
    cout << "El heroe tiene: "<< vida << " de vida"<<endl;

    return 0;
}
```

Salida en consola:



OPERADORES

Un operador es un símbolo que permite hacer interactuar a una o más variables modificándoles su estado, su valor o generando condiciones entre estas. Existen varios tipos de operadores. Por ejemplo, de momento nosotros utilizamos el = como asignar un valor a una variable, pues bien, el = es un operador y se utiliza precisamente para eso, asignar un valor a una variable, el = es un operador binario porque en su utilización intervienen dos operando, el de la izquierda y el de la derecha. En C++ tenemos algunos de los siguientes operadores:

Los operadores **aritméticos** operan números y el resultado es otro número.

Operador	Acción	Ejemplo	Resultado
-	resta	x = 5 - 3	x vale 2
+	suma	x = 5 + 3	x vale 8
/	división	x = 6 / 3	X vale 2
*	producto	x = 5 * 3	x vale 15
%	módulo	x = 5 % 2	x vale 1
--	decremento	x = 5; x --	x vale 4
++	incremento	x = 5; x++	X vale 6

El ++ y el -- son operadores unarios ya que sólo interviene un único operando, y se utilizan para incrementar o decrementar el valor de la misma variable. Es equivalente a decir x = x - 1 (x--) o bien x = x + 1 (x++).

Los operadores de **asignación** se utilizan para asignar un valor a la variable, ya sea sin modificar u realizando alguna operación aritmética en simultáneo.

Operador	Acción	Ejemplo	Resultado	Equivalente
=	Asignación básica	x = 6	x vale 6	
En los siguiente ejemplos, para x = 6				
+=	Asigna suma	x += 10	x vale 16	x = x + 10
/=	Asigna división	x /= 2	x vale 3	x = x / 2
*=	Asigna producto	x *= 3	x vale 18	x = x * 3
%=	Asigna módulo	x %= 5	x vale 1	x = x % 5

Los operadores **lógicos** se utilizan para comparar varios operandos y su resultado es siempre un valor booleano. Suponiendo la variable x = 5:

Operador	Relación	Ejemplo	Resultado
<	Menor que	x < 6	true
>	Mayor que	x > 10	false
<=	Menor o igual que	x <=5	true
>=	Mayor o igual que	x >= 8	false
==	Igual que	x == 5	true
!=	Distinto que	x != 5	false

Los operadores lógicos adquirirán muchísima importancia en las siguientes unidades. Veamos un ejemplo del uso de varios operadores:

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

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

    int vida = 100;
    int golpe = 5;
    int baya = 15;
    cout << "La vida del heroe es:" << vida <<endl;
    vida -= golpe;
    cout << "Oh no, el heroe fue golpeado y le resta " << golpe <<" de vida"<<endl;
    cout << "La vida del heroe es:" << vida <<endl;
    vida ++;
    cout << "Bien! El heroe junto un salmo de salud y que agrega 1 en vida"<<endl;
    cout << "La vida del heroe es:" << vida <<endl;
    vida = vida + baya;
    cout << "Bien! el heroe comio una baya energetica, su vida sube en "<<baya<<endl;
    cout << "La vida del heroe es:" << vida <<endl;
    bool vivo = vida > 0;
    cout << "_ Se encuentra el heroe con vida?: " << vivo <<endl;

    return 0;
}
```

La salida en consola es:

```
La vida del heroe es:100
Oh no, el heroe fue golpeado y le resta 5 de vida
La vida del heroe es:95
Bien! El heroe junto un salmo de salud y que agrega 1 en vida
La vida del heroe es:96
Bien! el heroe comio una baya energetica, su vida sube en 15
La vida del heroe es:111
_ Se encuentra el heroe con vida?: 1


<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

STRINGS (CADENAS)

Habíamos visto que las variables de tipo char sirven para guardar caracteres, pero no sirven para guardar cadenas. Es decir, la variable char *a* puede almacenar el valor 'a', 'z', '3' por ejemplo, pero no podrá almacenar la palabra "heroina", "enemigos" o "barra de energía". Recordar que los caracteres se escriben con comillas simples ' ' mientras las cadenas se escriben con comillas dobles " ", aun si la cadena tiene un único carácter, si va con comilla doble se considera cadena y no puede ser almacenada en una variable tipo char, es decir "a" es una cadena de un único carácter.
Para poder almacenar cadenas existe un tipo de dato llamado string. En realidad string es una clase (ya veremos esto más adelante) y no es un tipo de dato primitivo de C++, es por eso que lo mencionamos aparte. Pero desde hace ya muchos años fue incluida en el conjunto de elementos de C++ que se considera estándar. Esta serie de elementos se puede utilizar gracias a la incorporación de la línea **using namespace std;** que se encuentra encima de la función main en nuestro código. El uso de string es como cualquier otro tipo de dato, pero puede almacenar una cadena de caracteres.

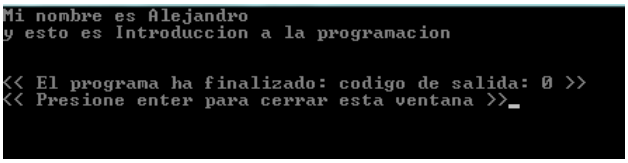
```cpp
#include<iostream>
using namespace std;

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

    string nombre = "Alejandro";
    cout<<"Mi nombre es "<<nombre<<endl;
    string materia = "Introduccion a la programacion";
    cout<<"y esto es "<<materia<<endl;

    return 0;
}
```

La salida en consola es:



La clase string tiene algunas funcionalidades muy útiles, pero las veremos más adelante.

COMENTARIOS

Existe la posibilidad de dejar comentarios en nuestro código, esto tiene una utilidad enorme ya que nos permite dejar notas para otras personas o para nosotros mismos o bien ocultar una porción de código mientras probamos alguna alternativa y no queremos borrarla.
Hay dos tipos de comentarios, los comentarios en línea y los comentarios en bloque. Los comentarios en línea se realizan con el operador // (barra barra) y sirven para comentar una única línea. Los comentarios en bloque sirven para comentar una porción de código y tiene dos operadores, el operador /* que abre el comentario y el operador */ que lo cierra.

Ejemplos:

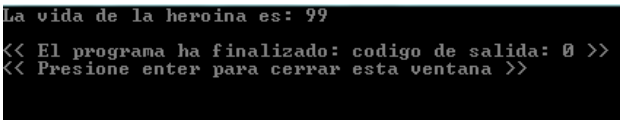
```
#include<iostream>
using namespace std;

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

    // La vida de la heroina comienza en 100
    int vida = 100;
    // Al pisar una hiedra venenosa se reduce en 1 en cada turno
    vida--;
    /* Si la vida llega a 0
    La heroina muere, en tal caso
    la vida debe volver a 100 */
    cout << "La vida de la heroina es: "<< vida;

    return 0;
}
```

La salida en consola es:



ENTRADAS Y SALIDAS

A lo largo del texto vimos que la instrucción **cout** sirve para mostrar algo en pantalla, a lo que denominamos **salida** ya que estamos sacando información de nuestro código al exterior, de la misma forma podemos ingresar información. Análogamente a la instrucción **cout** tenemos la instrucción **cin**, su uso es similar, pero los operadores no son el <<, sino el >> y para ingresar un valor debemos pasarle una variable para que lo aloje.
Ejemplo:

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

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

    // creamos una variable nombre
    string nombre;
    cout<< "Ingrese su nombre: ";
    // Pedimos que ingrese el nombre
    cin >> nombre;
    cout<<endl;
    cout<< "El nombre del personaje es: "<<nombre;

    int edad;
    cout<<endl;
    cout<< "Ingrese la edad: ";
    // Pedimos que ingrese la edad
    cin >> edad;
    cout<<endl;
    // no puedo escribir la ñ asi que escribo anios
    cout << nombre << " tu edad es de " << edad <<" anios";

    return 0;
}
```

La salida será:

```
Ingrese su nombre:
```

El cursor quedará esperando el tipeo del usuario. Tipeo “Alejandro” y presiono Enter.

```
Ingrese su nombre: Alejandro

El nombre del personaje es: Alejandro
Ingrese la edad:
```

Tipeo 100

```
Ingrese su nombre: Alejandro

El nombre del personaje es: Alejandro
Ingrese la edad: 100

Alejandro tu edad es de 100 anios

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

PALABRAS RESERVADAS

La siguiente es una lista de palabras reservadas en C++ que sólo puede utilizarse para su función específica

asm	continue	float	new	signed	try
auto	default	for	operator	sizeof	typedef
break	delete	friend	private	stact	union
case	do	goto	protected	struct	unsigned
catch	double	if	public	switch	virtual
char	else	inline	register	template	void
class	enum	int	return	this	volatile
const	extern	long	short	trhow	while