Estructura general de los programas

Para comenzar con un programa cualquiera en C++, es necesario disponer de un IDE y un compilador de C++ instalado y configurado.

Como se explica en el Campus Virtual, Codeblocks es un producto de software que luego de instalarlo nos provee de un entorno de desarrollo listo para utilizar y compilar.

En el transcurso de la materia, todos los programas que realicemos compartirán este código fuente base que será nuestro punto de partida para empezar a codificar.

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

int main(){

 return 0;
}
```

Código 1 · Código fuente que utilizaremos como base para todos los programas

	La línea 1 realiza la inclusión de una biblioteca. Con esto agregamos una serie de funcionalidades extra para nuestros programas tales como la facultad de ingresar por teclado, mostrar por pantalla, esperar una tecla cualquiera para continuar, etc. La línea 2 permite el uso de cin y cout indicando que se hará uso del espacio de nombre std. Al principio puede parecer confuso pero en el transcurso de la carrera esto será muy sencillo de asimilar y utilizar.

Programa principal o main()

En C++ podemos hacer diferentes subprogramas o funciones, pero el único obligatorio para poder hacer una aplicación es la función main. En ella se escribirán todas las líneas con instrucciones que sean necesarias para que nuestro programa resuelva lo que queremos.

Case-sensitive

Algo importante a acostumbrarse es que un lenguaje de programación es extremadamente estricto con la sintaxis. Específicamente C++ tiene la característica que

hace diferencia entre las minúsculas y las mayúsculas. Por ejemplo, para este lenguaje de programación las variables minumero, MiNumero y MINUMERO son distintas. Que un texto sea analizado de manera distintas en función de si está en mayúsculas o minúsculas se denomina "case-sensitive". Como programadores es fundamental que se acostumbre a esta restricción ya que será clave al momento de codificar.

Declaración de variables

A diferencia de como sería en una representación gráfica (diagrama) de la resolución de nuestro programa (algoritmo), cuando programemos en C++ necesitamos declarar todas las variables que vamos a utilizar. Las variables pueden representar cosas distintas y por eso decimos que son de distinto **tipo**. El tipo de dato nos indica qué información se puede guardar en nuestras variables, por ejemplo:

Tipo C/C++	Castellano	Descripción	Ejemplo

int	entero	Un número que no tiene la capacidad de guardar valores decimales.	`10`
			`-4`
			`0`

float	real	Un número que tiene la capacidad de guardar valores decimales.	`5.5`
			`-5.6`
			`100`
			`0.00025`

char	carácter	Un carácter cualquiera. Puede ser número, letra, símbolo, etc.	`A`
			`f`
			`@`
			`4`

bool	booleano	Un valor de verdad.	`1`
			`0`
			`true`
			`false`

string	cadena	Un texto que puede contener letras, números y caracteres especiales (sólo puede ser utilizado en C++)	`Jerry`
			`George`
			`Elaine`
			`Kramer`

Tabla 1 · Tipos de datos en C/C++

```
int num, edad, cantidad;
char genero, forma_de_pago;
float importe, promedio;
string apellido;
```

Código 2 · Ejemplos de declaración de variables en C/C++

Los nombres de las variables deben cumplir una serie de reglas. Las mismas son:

- No pueden empezar con número ni contener símbolos, a excepción del guión bajo. Esto incluye a caracteres como la ñ y las letras con tilde que no son permitidos.
- No pueden contener espacios.
- No pueden ser iguales a términos utilizados por el lenguaje para otras cosas (palabras reservadas del lenguaje), tales como: main, int, char, float, bool, if, switch, case, for, while, break, continue,, etc

✖	int 2num; // Empieza con un número
	float importe total; // Tiene un espacio
	char while; //while es una palabra reservada del lenguaje
	float %aprobados; //Utiliza un símbolo no válido para declarar variables

¡Error! · Ejemplos de nombres no válidos para variables

Ingreso de datos

El ingreso de datos se realiza mediante el uso del comando cin. Esto permitirá que el usuario ingrese por teclado un valor y luego de presionar Enter, este sea asignado a una variable. La sintaxis es la siguiente:

```
cin >> numero;
```

Código 3 · Uso de cin para ingresar un dato a una variable llamada numero.

También es posible resolver en una misma línea varios ingresos a la vez. Esto ejecuta el comando cin por cada una de las variables que se indiquen.

```
cin >> legajo >> nota_practica >> nota_teoria;
```

Código 4 · Uso de cin para ingresar datos a varias variables.

En el código anterior, se ingresan primero el legajo, luego la nota de práctica y por último la nota de teoría en tres ingresos separados. Cada ingreso debe ser confirmado con la tecla Enter aunque también pueden ser separados por un espacio y un Enter.

Salida de datos

La salida de datos se realiza mediante el comando cout. Esto permitirá que el usuario muestre por pantalla un valor. Este valor puede ser lo que contenga una variable o bien un texto literal.

```
cout << numero;
```

Código 5 · Muestra por pantalla el contenido de la variable numero.

```
cout << "El resultado de la operación es: ";
```

Código 6 · Muestra por pantalla el texto *El resultado de la operación es:* de manera literal.

```
cout << "El resultado de la suma es: " << suma;
```

Código 7 · Muestra por pantalla el texto El resultado de la operación es: y luego, a su lado, el contenido de la variable suma.

También, es posible que deseemos mostrar por pantalla varias cosas a la vez y que las mismas aparezcan separadas por espacios verticales. Para ello utilizaremos el siguiente elemento en un cout.

```
cout << endl;
```

Código 8 · Muestra por pantalla un salto de línea.

Cada uno de ellos representa un espacio vertical o "línea vacía". Podemos poner varios para que se apliquen varias líneas de distancia entre contenidos, por ejemplo, si queremos tres líneas de distancia utilizaremos:

```
cout << endl << endl << endl;
```

Código 9 · Muestra por pantalla tres saltos de línea.

La importancia del endl radicará en cómo se verá por pantalla el resultado. Por ejemplo, el siguiente código mostraría un resultado correcto pero desprolijo:

```
cout << "El resultado de la suma es : " << resultado;
cout << "El resultado de la resta es: " << resultado2;
```

Salida

```
El resultado de la suma es : 4349El resultado de la resta es: 3423
```

En cambio este código permitirá que se visualice de una mejor manera

```
cout << "El resultado de la suma es : " << resultado;
cout << endl << endl;
cout << "El resultado de la resta es: " << resultado2;
```

Salida

```
El resultado de la suma es : 4349

El resultado de la resta es: 3423
```

Operadores

Los operadores nos permiten hacer todo tipo de operaciones con la información que contenemos en variables o en constantes. Por ejemplo, con ellos podemos asignar información a variables, realizar operaciones matemáticas, comparar valores entre sí, etc.

Operador de asignación

Permite asignar un valor a una variable. El valor a asignar puede provenir de una constante, el resultado de una operación aritmética o de otra variable.

```
int edad;
char genero;
float altura;
int res;
int res2;
string personaje;
edad = 20;
genero = 'F';
altura = 1.84;
res = ((5*7)+4)/2;
res2 = res;
personaje = "Pennywise";
```

Código 10 · Ejemplo de asignación de valores de distintos tipos de datos a variables.

En el ejemplo anterior se asignan los valores 20 a la variable edad, la letra F a la variable género, el valor 1.84 a la variable altura y el texto (cadena) Pennywise a la variable

personaje. También se asigna el resultado de una operación aritmética a la variable res y luego el contenido de esta variable a la variable res2.

Presten especial atención en como la asignación de un carácter a una variable char debe hacerse con comillas simples, la asignación de un texto a una variable de tipo string debe hacerse con comillas dobles y como el separador decimal en las variables de tipo float es el punto. Esto es así porque así lo determina el lenguaje. Es parte de la sintaxis.

Recordar que del lado izquierdo del operador de asignación debe ir obligatoriamente una variable ya que la instrucción representa que la variable que se encuentra a la izquierda le es asignado el valor que se encuentra a la derecha.

Operadores matemáticos

Permiten hacer todo tipo de operaciones matemáticas básicas. Tales como suma, resta, producto y división. De la división podremos obtener, si lo deseamos, el cociente y el resto. Los símbolos utilizados para cada operación son los siguientes:

Símbolo	Operación
+	Suma
-	Resta
*	Producto
/	Cociente o División real
%	Resto de la división entre enteros

Tabla 2 · Operadores matemáticos.

Comentarios de código

Los comentarios son secciones de código que no son analizadas por el compilador. Es decir, no generarán errores ya que no evaluará su sintaxis. Estos comentarios tienen dos propósitos, el primero de ellos es cancelar un fragmento de código fuente para evitar su compilación, y por ende, su posterior ejecución.

Otro de los propósitos del uso de comentarios es para literalmente dejar comentarios en el código fuente. Esto es de gran importancia ya que permite documentar nuestro trabajo y dejar instrucciones de lo que hace nuestro programa o fragmento del mismo.

Comentario de una línea

Permiten realizar un comentario que afecta a una sola línea de nuestro código fuente. El mismo puede comenzar al principio de la línea cancelando completamente la misma o bien a la derecha de una instrucción en la línea.

El mismo se realiza poniendo la combinación de caracteres //

```
int cant;
float pu, subtotal, total;
cin >> cant;
cin >> pu;
subtotal = pu * cant;
total = subtotal * 0.9; // Se aplica un 10% de descuento
// cout << "Subtotal: $ " << subtotal << endl;
cout << "Total    : $ " << total << endl;
```

Código 11 · Ejemplo de comentarios de una línea

Comentario de varias líneas

Permiten realizar un comentario que afecta a varias líneas de nuestro código fuente. Es de gran utilidad para cancelar un gran fragmento de nuestro programa sin la necesidad de eliminar las líneas de manera directa o para dejar un comentario extenso en nuestro código.

El mismo se realiza poniendo la combinación /* para iniciar el comentario y */ para finalizarlo.

```
/*
  El siguiente programa ingresa la cantidad y el precio unitario de un
producto y calcula el total aplicando un 10% de descuento. Mostrando el
subtotal y el total del importe.
*/
int cant;
float pu, subtotal, total;
cin >> cant;
cin >> pu;
subtotal = pu * cant;
total = subtotal * 0.9;
/*
cout << "Subtotal: $ " << subtotal << endl;
cout << "Total    : $ " << total << endl;
*/
cout << subtotal << " " << total << endl;
```

Código 12 · Ejemplo de comentarios de varias líneas