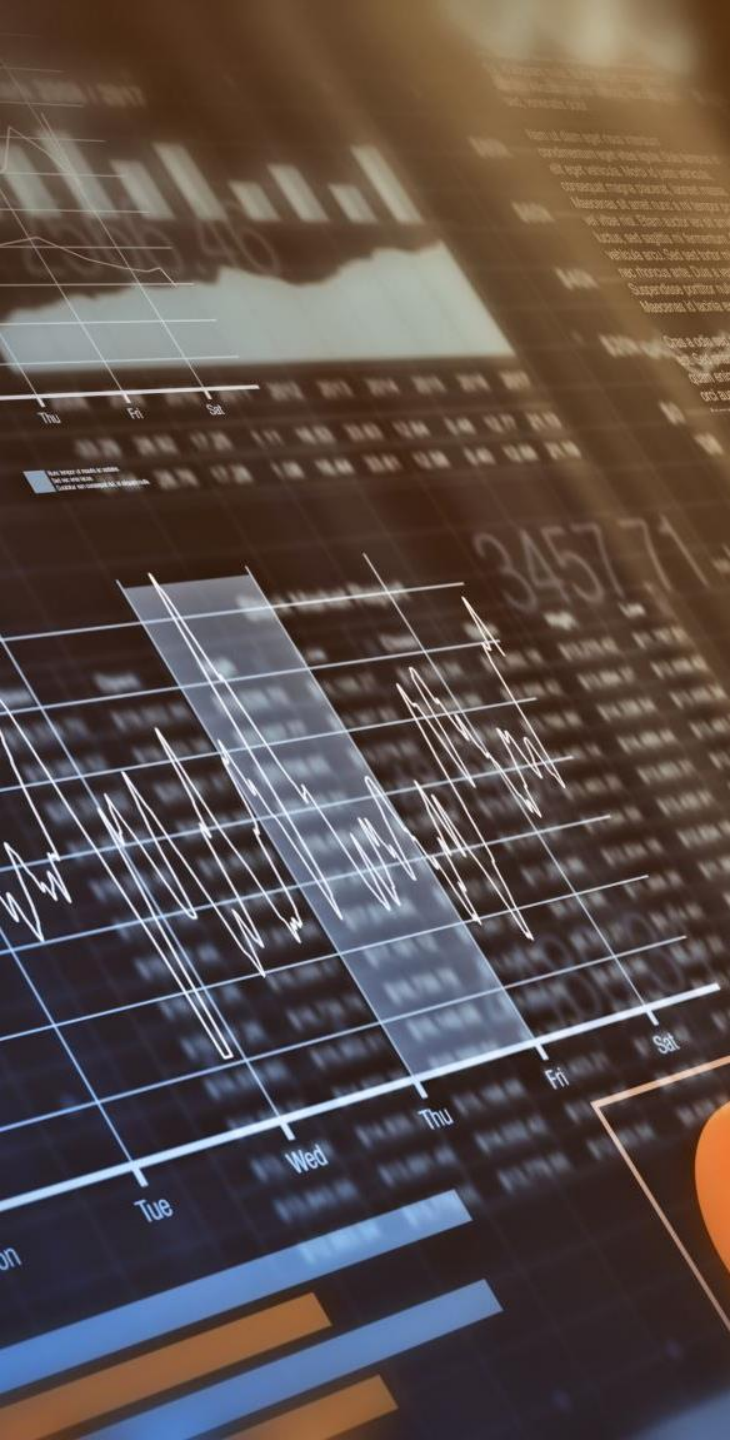


PROGRAMACIÓN AVANZADA

ING. INDUSTRIAL

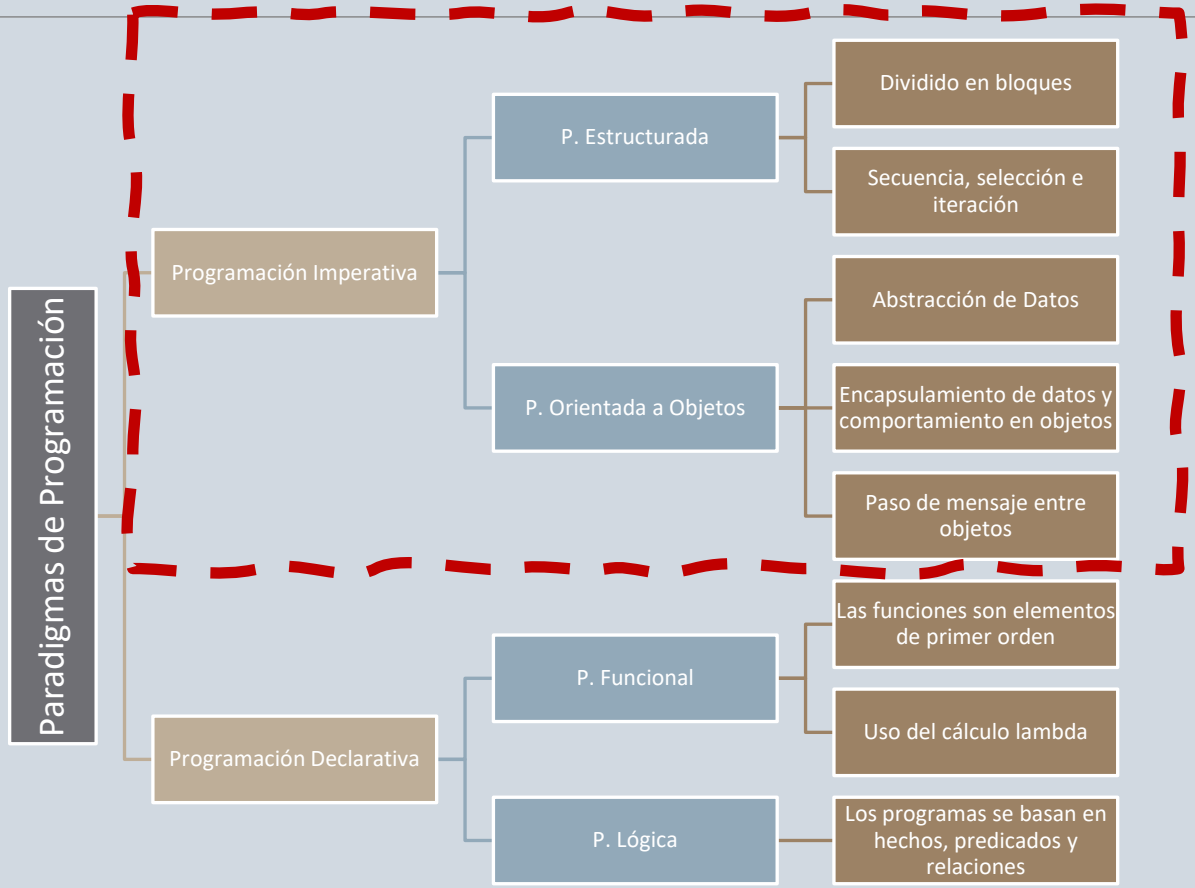
FAC. DE INGENIERÍA

UNIVERSIDAD NACIONAL DE JUJUY

Introducción C++

Estructura de un programa en C++.Elementos de programas C++. Tipos de datos de C++. Expresiones. Entrada y Salida. Estructuras de control

Enfoques de Programación



Programación Estructurada

Enfoque disciplinado que permite escribir programas estructurados, utilizando las siguientes tres estructuras de control bien definidas:

- **Secuencial** (asignación, lectura, escritura)
- **Decisión o selección** (simple, doble, múltiple)
- **Repetición** (repita-mientras, hacer-mientras, repita-para)

Los programas estructurados son fáciles de probar, depurar y modificar.

La programación se divide en **bloques** (procedimientos y funciones) que pueden o no comunicarse entre sí.

Permite **reutilizar** código programado y otorga una mejor compresión de la programación..

Programación Orientada a Objetos

En el paradigma orientado a objetos, el programa se organiza como un conjunto finito de **objetos** que contienen datos y operaciones (funciones miembro en C++) que llaman a esos datos y que se comunican entre sí mediante **mensajes**.

Lenguaje de Programación C++

C++ el lenguaje de la ingeniería

Definición:

- C++ es un lenguaje de programación de alto nivel, potente y flexible, ampliamente utilizado en la industria para desarrollar aplicaciones de software de alta performance.

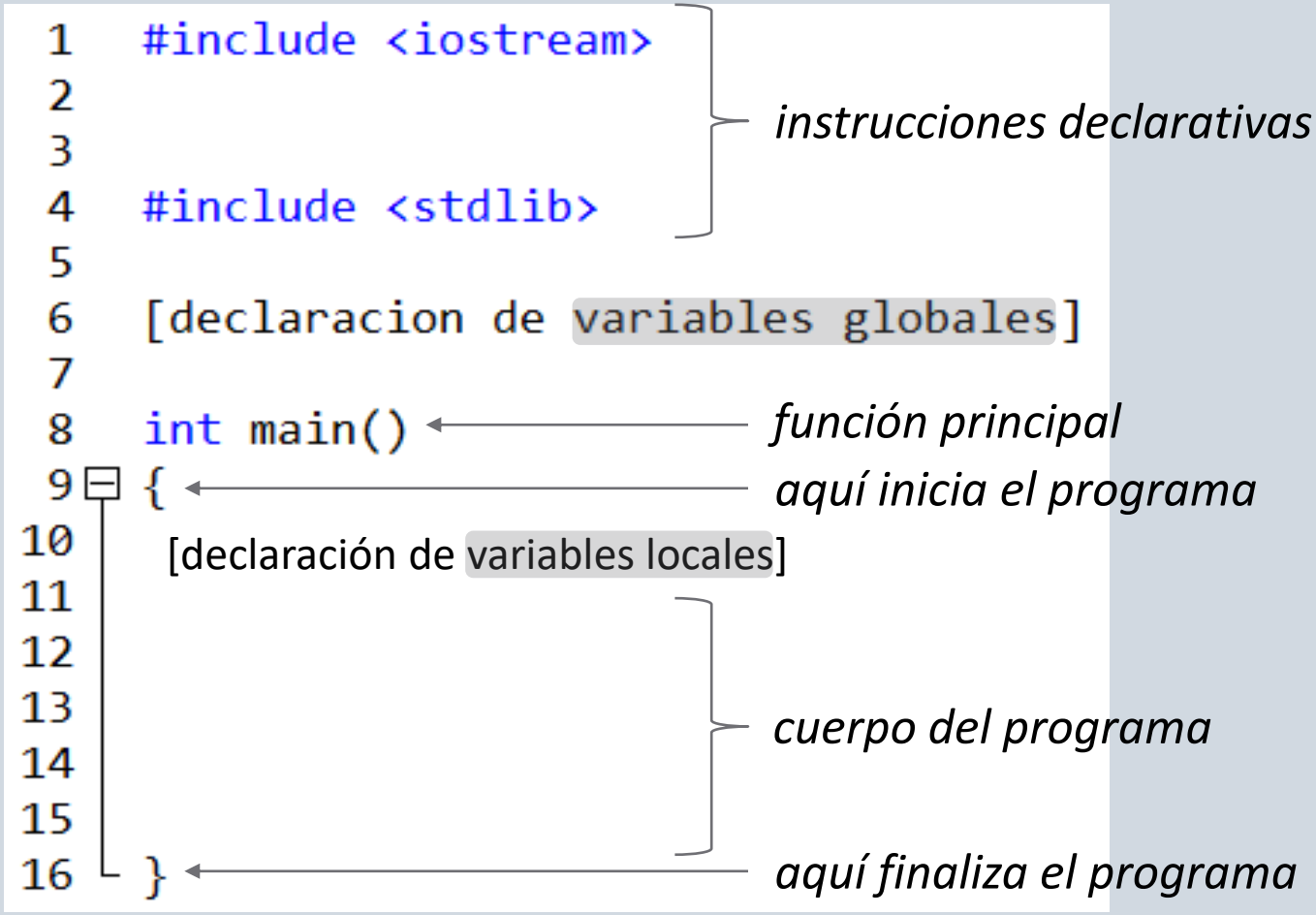
Versatilidad:

- C++ permite construir desde pequeños programas hasta sistemas operativos y videojuegos. En ingeniería industrial, su versatilidad se traduce en:
  - **Simulación:** Modela sistemas complejos y realiza análisis "qué pasaría si".
  - **Optimización:** Encuentra la mejor solución a problemas industriales.
  - **Control de calidad:** Automatiza pruebas y garantiza la calidad de los productos.
  - **Robótica:** Programa el movimiento y la interacción de robots.

Características Generales de C++

- Lenguaje híbrido: Programación Estructurada/POO
- Economía de las expresiones
- Abundancia de operadores y tipos de datos
- Codificación en alto nivel y bajo nivel
- No está orientado a ningún área en especial
- Producción de código objeto altamente optimizado

Estructura de un programa en C++



Estructura de un programa en C++

#include<librería_solicitada>

- con esta directiva de preprocesador se incorporan las bibliotecas de funciones que se van a utilizar
- Las librerías más utilizadas son:
  - <iostream> contiene las funciones para ingresar y mostrar datos
  - <math.h> contiene las funciones matemáticas comunes
  - <time.h> contienes las funciones para tratamiento y conversión entre formatos de fecha y hora
  - <stdio.h> contiene los prototipos de las funciones, macros y tipos para manipular datos de E/S
  - <stdlib.h> contiene tipos, macros y funciones para la conversión numérica, generación de memoria y tareas similares
  - <string.h> contiene los prototipos de las funciones y macros de clasificación de caracteres

Estructura de un programa en C++

main() { … }

- Es la función principal de un programa en C++.
- Todo programa en C++ comienza con una función main() que es única.
- Los paréntesis que le siguen contienen lo que se le va a mandar a la función
- Entre las llaves { } se incluirá el **bloque de instrucciones/sentencias** que resuelve el problema.
- Las sentencias:
  - definen la lógica de un programa o subprograma
  - manipulan los datos para producir el resultado deseado por el usuario del programa

Estructura de un programa en C++

Sentencias

- definen la lógica de un programa o subprograma
- manipulan los datos para producir el resultado deseado por el usuario del programa
- En programación estructurada, se tendrán sentencias que permiten
  - Estructura secuencial
  - Estructura selectiva
  - Estructura repetitiva

Estructura de un programa en C++

Dato

- Un dato es **cualquier valor o pieza de información** que un programa puede utilizar.
- Estos datos pueden ser números, texto, fechas, imágenes, etc.
- Para que la computadora pueda trabajar con estos datos, es necesario definir su tipo y cómo se representarán en la memoria.
- Los datos en un programa se pueden representar a través de
  - Variables
  - Constantes

Estructura de un programa en C++

Variable

- Una variable es un espacio reservado en la computadora para contener valores que pueden **cambiar** durante la ejecución de un programa.
- Toda variable tiene un **tipo de dato**, el cual determina los valores que se pueden representar y las operaciones que se pueden realizar con ellos.
- Todo programa puede tener
  - **Variables globales**. Una **variable global** es válida para todas las funciones que componen el programa.
  - **Variables locales.** Una **variable local** es aquella cuyo ámbito se restringe a la función que la ha declarado se dice entonces que la variable es local a esa función.

Estructura de un programa en C++

Constantes

- Una constante tiene las mismas características que una variable excepto el hecho de que su valor asignado no puede ser cambiado durante la ejecución del programa.
- Toda constante tiene un nombre y un valor predefinido

Estructura de un programa en C++

Comentarios

- Un comentario es cualquier información añadida al archivo fuente del programa e ignorada por el compilador.
- En C y C++ los comentarios de varias lineas comienzan por **/*** y terminan con la secuencia ***/**.
- En C++ se define **una línea** de comentario comenzando con una doble barra inclinada (**//**).

Estructura de un programa en C++

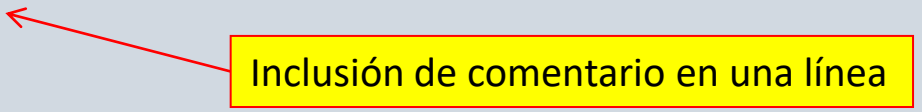
```
#include<iostream>
//programa que muestra un saludo clásico

using namespace std;

int main(){
   cout << “Hola mundo!” << endl;
   return 0;
}
```

Estructura de un programa en C++

```
#include<iostream>
//programa que muestra un saludo clásico

using namespace std;

int main(){
   cout << "Hola mundo!" << endl;
   return 0;
}
```

Estructura de un programa en C++

```
#include<iostream>
//programa que muestra un saludo clásico

using namespace std;

int main(){
   cout << "Hola mundo!" << endl;
   return 0;
}
```



Estructura de un programa en C++

```
#include<iostream>
//programa que muestra un saludo clásico

using namespace std;

int main(){
   cout << "Hola mundo!" << endl;
   return 0;
}
```

Uso global de la librería estándar

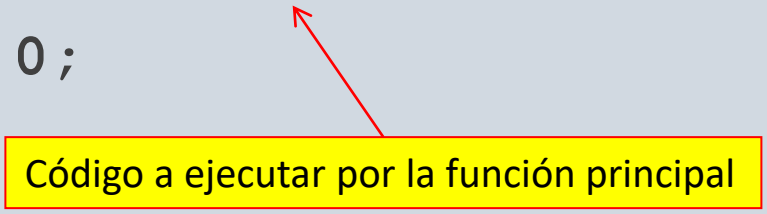
Estructura de un programa en C++

```
#include<iostream>
//programa que muestra un saludo clásico

using namespace std;

int main(){
   cout << "Hola mundo!" << endl;
   return 0;
}
```

Declaración de la función principal

Estructura de un programa en C++

```
#include<iostream>
//programa que muestra un saludo clásico

using namespace std;

int main(){
   cout << "Hola mundo!" << endl;
   return 0;
}
```



Elementos de un programa en C++

Otros elementos básicos de un programa en C++ son:

- identificadores
- palabras reservadas
- signos de puntuación
- separadores

Identificadores

Un *identificador* es una secuencia de caracteres, letras, dígitos y subrayados (_) que comienza siempre por un carácter. Las letras mayúsculas y minúsculas son diferentes. Pueden tener cualquier longitud, pero el compilador ignora a partir del 32. No pueden ser palabras reservadas.

Ejemplos de identificadores:

Nombre_Alumno	LetraIndice	Dia	Mayor	menor
Fecha_Compra	Alfa	habitacion24	i	j

Palabras reservadas

C++, como la mayoría de los lenguajes, tiene reservados algunos identificadores con un *significado especial*, que sólo pueden ser usados con ese cometido.

Una *palabra reservada*, tal como void (ausencia de tipo, o tipo genérico), es una característica del lenguaje C++.

Una *palabra reservada* **no** se puede utilizar como nombre de variable, de constante, de objeto o de función.

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

Palabras reservadas

Signos de Puntuación y Separadores

Todas las sentencias de C++ deben terminar con un punto y coma (;).

Los separadores son espacios en blanco, tabulaciones, retornos de carro y avances de línea.

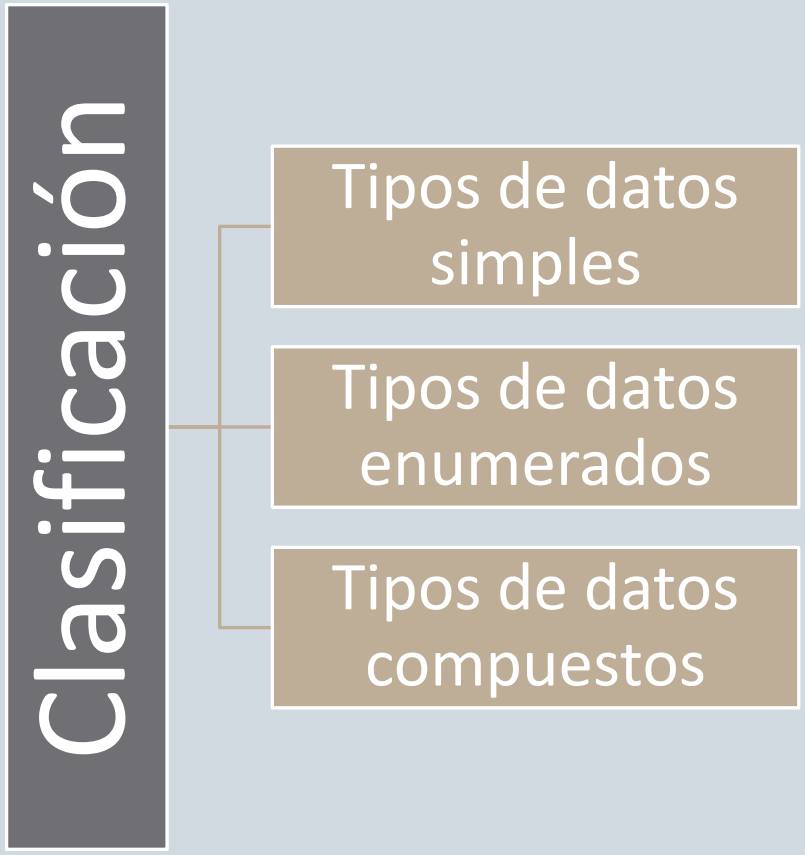
Otros signos de puntuación son:

!	-	]
%	+	\
^	=	;
&	{	‘
*	}	:
(	~	<
)	[	>
?	,	.
	/	

Tipos de Datos en C++

UNIDAD 1

Tipos de datos en C++



Tipos de datos Simples en C++

C++ es un lenguaje de programación que hereda muchos conceptos del lenguaje C, es un ***lenguaje compilado y fuertemente tipado***, lo que significa que en las variables con las que trabajamos hay que indicar el tipo del dato que van a guardar cuando se declaran, lo que puede hacer que tengamos problemas y se generen errores.

Tipos de Datos Simples en C++

Nombre	Descripción	Tamaño*	Rango de valores*
char	Carácter o entero pequeño	1byte	con signo: -128 to 127 sin signo: 0 a 255
short int (short)	Entero corto	2bytes	con signo: -32768 a 32767 sin signo: 0 a 65535
int	Entero	4bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
long int (long)	Entero largo	8bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
bool	Valor booleano. Puede tomar dos valores: verdadero o falso	1byte	true o false
float	Número de punto flotante	4bytes	3.4e +/- 38 (7 digitos)
double	De punto flotante de doble precisión	8bytes	1.7e +/- 308 (15 digitos)
long double	Long de punto flotante de doble precisión	8bytes	1.7e +/- 308 (15 digitos)

Tipos de Datos enumerados en C++

Los tipos enumerados son un mecanismo usado en C++ que nos permite agrupar ***constantes simbólicas***, por ejemplo, los días de la semana, los meses del año, los colores primarios, etc. etc.

Ejemplo:

enum días {lunes, martes, miércoles, jueves, viernes, sábado, domingo};

Tipos de datos compuestos en C++

Los tipos de datos compuestos/complejos son aquellos que permiten guardar una cantidad de elementos bajo un único identificador.

Los tipos compuestos clásicos son los arreglos (vectores, matrices) y los registros.

Declaración de variables en C++

Es preciso declarar las variables antes de utilizarlas. La sintaxis de declaración es:

tipo nombre-de-variable[=valor];

lo que está entre corchetes es opcional, se puede o no dar el valor a la variable al mismo tiempo que se la declara. Además, se puede declarar varias variables del mismo tipo separándolas con comas. Si se quisiera declarar variables como las básicas, sería así:

```
1. int T=1401, variable=1, a;
2. float pi=3.14, decimal, otra=0.23;
3. char letra='C', mas;
4. bool dicho=true, bandera=false;
```

Nota: *después de declarar una serie de variables de un mismo tipo hay que poner un* **;**

Modificadores de tipo de dato

Los tipos de datos tienen un tamaño definido en bits y un rango, los modificadores permiten modificar a los tipos de datos de acuerdo con su rango. Ellos son:

- **short** (corto) afecta a int.
- **long** (largo) afecta a int y double.
- **signed** (con signo) afecta a char e int. Hace referencia a – o +
- **unsigned** (sin signo) afecta a char e int. Considera todos positivos

Los ejemplos de uso son muy simples, por ejemplo

- si se requiere un entero corto se usa ***short int***.
- si se requiere un entero largo sin signo se usa ***unsigned long int.***

Los usos que se le puede dar a los modificadores pueden ser muchos, p.e. ***cálculos exactos, programas con características especiales***, etc.

Entrada y Salida

UNIDAD 1

Entrada y Salida en C++

Las bibliotecas estándar de C++ proporcionan un amplio conjunto de capacidades de entrada/salida (E/S).

C++ utiliza E/S a prueba de tipos. Cada operación de E/S se realiza automáticamente en una forma sensible con respecto al tipo de datos

Flujos: La E/S de C++ se da en flujos de bytes. Un flujo es simplemente una secuencia de bytes.

- **cin y cout:** cin es el flujo de entrada estándar que normalmente es el teclado y cout es el flujo de salida estándar que por lo general es la pantalla. Para utilizar cualquiera de estos flujos debe incluirse **iostream** que es el archivo de encabezado del flujo de **entrada/salida**

Salida de datos a consola

- En C++ se puede usar el flujo **`cout`** para imprimir información en la pantalla, utilizando el operador de salida de flujo **<<**
- Se puede mostrar: *una cadena de texto, el contenido de una variable, el resultado de una expresión aritmética, etc*.
- La constante **`endl`** imprime un retorno de carro. También puede utilizarse "**\n**".

- Ejemplos:

```
cout<<"Lenguaje C++";
cout<<"El valor de x es "<<x<<endl;
cout<<"El area del circulo es "<<3.14159*r*r;
cout<<endl<<"\n"<<endl;
```

Entrada de datos desde consola

- El flujo de entrada **`cin`** permite introducir datos desde el teclado y almacenarlos en variables previamente declaradas.
- Ejemplo:

```
int main() {
  int x;
  cout << "Ingrese un numero entero: ";
  cin >> x;
  cout << "El numero que ingresaste es: ";
  cout << x << endl;
}
```

Expresiones

UNIDAD 1

Asignación de variables

- Para asignar un valor a una variable se utiliza el operador **=**
- Es posible asignar valores al momento de declarar las variables;

- Ejemplos:

```
float a = 3.4, b = 5.6, c = -1.7;
int x;
x = 10;
```

Expresiones aritméticas

Una expresión aritmética es aquella secuencia conformada por operadores aritméticos y operandos.

Una expresión aritmética puede unaria o binaria, de acuerdo a la cantidad de operando requeridos.

- -1
- 6*5

Operadores aritméticos

- En C/C++ se pueden formar **expresiones aritméticas** con los operadores comunes:

  \+ (suma), - (resta), * (multiplicación), / (división), **%** (resto)

- Debe tenerse siempre en cuenta que los operadores actúan de acuerdo con el tipo de dato de los operandos involucrados. Ejemplo:

```
x = 5 / 2;          // el resultado es 2
x = 5.0 / 2;        // el resultado es 2.5
```

- C/C++ no cuenta con un operador de potencia, pero se puede utilizar la función `pow()` de la librería math.h

Operadores Incrementales

C también tiene el *operador incremental* unario, **++**, y el *operador decremental* unario **--** que mostramos a continuación.

Operadores	Expresión de muestra	Explicación
++	++a	Se incrementa a en 1 y a continuación se utiliza el nuevo valor de a en la expresión en la cual resida a
++	a++	Utiliza el valor actual de a en la expresión en la cual reside a, y después se incrementa a en 1
--	--b	Se decrementa en 1 y a continuación se utiliza el nuevo valor de b en la expresión en la cual reside b
--	b--	Se utiliza el valor actual de b en la expresión en la cual reside b, y después se decrementa a b en 1

Expresiones booleanas

Una **expresión booleana** es aquella que después de evaluarse puede tomar uno de dos posibles resultados: **verdadero** o **falso.**

En lenguaje C/C++, los valores verdadero y falso se representan, respectivamente, con los números 1 y 0.

Sin embargo, cualquier valor distinto de cero es considerado como “verdadero”.

Operadores de comparación

- Una manera de formar expresiones booleanas es utilizando los **operadores de comparación**:

a **==** b - igualdad
a **<** b - menor a
a **>** b - mayor a
a **!=** b - distinto de
a **<=** b - menor o igual a
a **>=** b - mayor o igual a

Operadores booleanos

- Las expresiones booleanas pueden combinarse mediante los **operadores booleanos** clásicos:

&&	- conjunción (and)
||	- disyunción (or)
!	- negación (not)

- ***Nota 1:*** estos operadores difieren de los operadores lógicos binarios: &, |, ~
- ***Nota 2:*** en muchos casos, los operadores && y || pueden reemplazarse, respectivamente, por el producto y la suma.

Práctica #1

Escriba un programa que pida al usuario la base y la altura de un rectángulo y calcule y muestre su superficie.