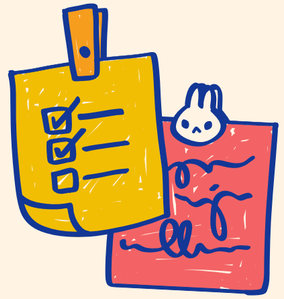
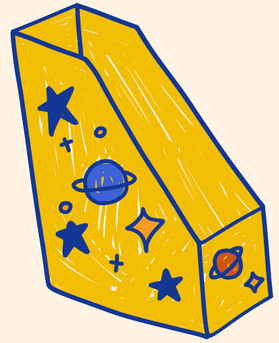






Curso de C++ básico

Programa de Tecnología en Cómputo (PROTECO)
Intersemestral 2023-1

PROTECO

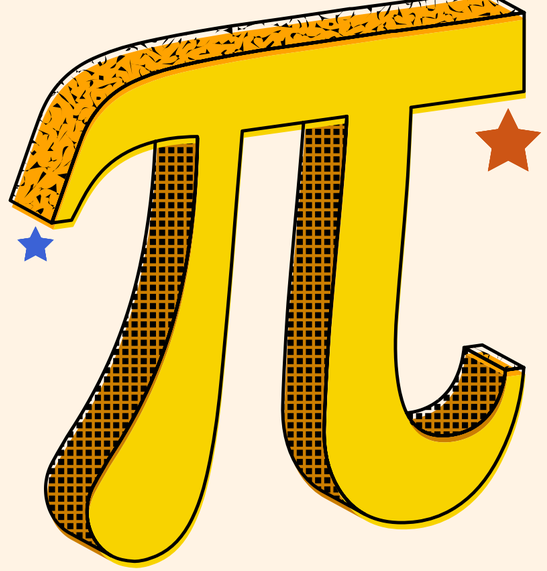
Tipos de datos en C++

Nombre	Descripción	Tamaño*	Rango de valores*
char**	Carácter o entero pequeño	1 byte	con signo: -128 a 127 sin signo: 0 a 255
short int (short)**	Entero corto	2 bytes	con signo: -32768 a 32767 sin signo: 0 a 65535
int**	Entero	4 bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
long int (long)**	Entero largo	4 bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
bool	Valor booleano. Puede tomar dos valores: Falso o verdadero	1 byte	true o false

Nombre	Descripción	Tamaño*	Rango de valores*
Float	Número de punto flotante.	4 bytes	3.4e +/- 38 (7digitos)
double	De punto flotante de doble precisión.	8 bytes	1.7e+/-308 (15 digitos)
long double	Long de punto flotante de doble precisión	8 bytes	1.7e+/-308 (15 digitos)

***Estos valores se toman de un estándar.**

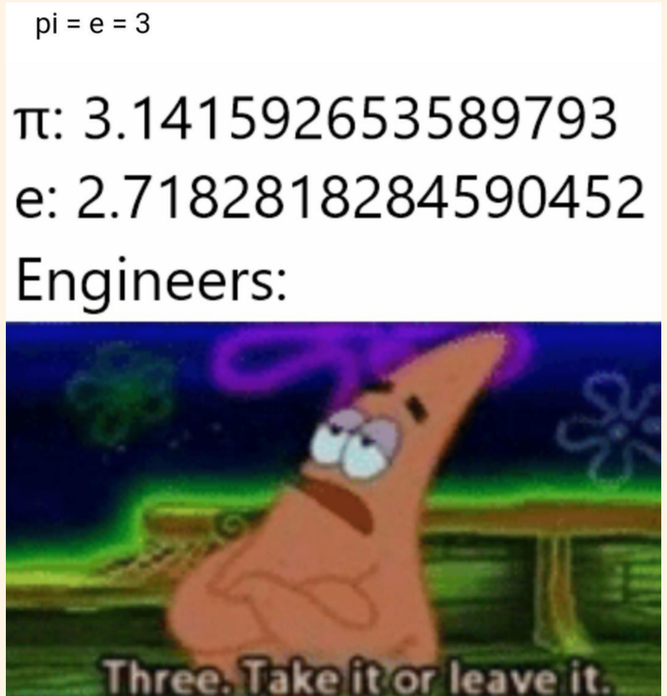
****El modificador unsigned agrega el rango de los negativos a la parte positiva, permitiendo almacenar únicamente números positivos.**



Variables y constantes.

Constante: Algo que no puede cambiar (un valor). Por ejemplo:

- Valor de número de PI.
- Valor de número de Euler.
- Valor de la gravedad.



Variable: Las variables son una herramienta utilizada en la programación para almacenar y manipular datos relevantes para la ejecución de un programa.
. Por ejemplo:

- Al hervir agua de 21°C cambia a 95°C.
- La estatura.

Estas se conforman por:

- Un espacio en memoria (RAM).
- Un nombre simbólico (identificador).
- Un dato o valor.

Variables y constantes.

Para **declarar (crear) una variable e**n C++ utilizamos la siguiente sintaxis:

<tipo> <identificador>;

Para **asignar un valor a una variable** en C++ utilizamos la siguiente sintaxis:

<identificador> = <valor>;

Para **declarar y asignar un valor a una variable** en C++ utilizamos la siguiente sintaxis:

<tipo> <identificador> = <valor>;

Para acceder al valor de una variable, basta con escribir su identificador.

Buenas prácticas

- Únicamente se pueden utilizar combinaciones de letras, números y guiones bajos para nombrar variables y funciones.
- No puede haber dos variables o funciones con el mismo identificador.
- El identificador de una variable no puede ser una palabra reservada.
- Todos los identificadores de variables y funciones deberían empezar con una letra minúscula.
- Si el identificador de una variable o función está compuesto por dos o más palabras, éstas se deben separar con **guiones_bajos**.
- Los identificadores y tipos de las variables deberían ser representativos de lo que almacenan.

Variables y constantes.

Para **declarar (crear) una constante** en C++ utilizamos la siguiente sintaxis:

const <tipo> <identificador> = <valor>;

Para **declarar (crear) una constante** en C++ **fuera del main** utilizamos la siguiente sintaxis:

#define <identificador> <valor>;

Flujo Estándar.

Recordando lo de ayer.

Los flujos son canales de comunicación interconectados mediante los cuales los programas pueden interactuar con su entorno durante su ejecución.
Hay tres tipos de flujos estándar:

- Entrada estándar (stdin), suele ser el teclado.
- Salida estándar (stdout), suele ser el monitor.
- Error estándar (stderr), también suele ser el monitor.

Manejo de flujos estándar en C++

En C++ se pueden utilizar los operadores de inserción (<<) y extracción (>>) de flujo junto con los objetos cin, cout y cerr para manipular la entrada, la salida

Estos objetos pertenecen al espacio de nombres std y se encuentran en la cabecera (o biblioteca) iostream. Su sintaxis de uso es la siguiente:

std::cin >> *<variable_1>* ***>> ... >>*** *<variable_n>;*

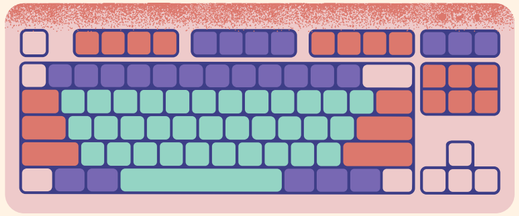
std::cout << *<variable_1>* ***<< ... <<*** *<variable_n>;*

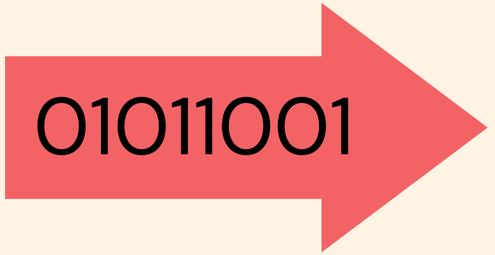
std::cerr << *<variable_1>* ***<< ... <<*** *<variable_n>;*

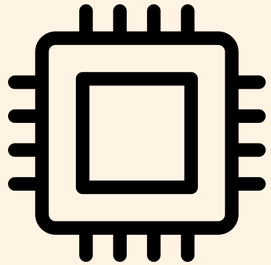
A diferencia de C, C++ interpreta automáticamente los tipos de dato de las variables al pasarlas por algún flujo.

¿Hacia dónde van los paréntesis angulares?

std::cin >>







std::cout <<