



# Lenguaje C++

Programación orientada a objetos



# INTEGRANTES

- Juan David Ortiz Cano 1152298
- Alison Brigitte Martínez Machado 1152299
- María Fernanda Corzo Castro 1152300
- Jorge Andrés Forero Serrano 1152328
- Linda Valentina Bohórquez Hernández 1152334

# Historia



1980:

La intención de su creación fue extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos.

1982


Se sumaron los paradigmas de programación estructurada y programación orientada a objetos.

1983:

El nombre C++(incremento de C) fue propuesto por Rick Mascitti.

1984:

Se empieza a usar la directiva #include, así mismo los espacios de nom



1985:

- Caracteres: char (también es un entero), wchar\_t
- Enteros: short, int, long, long long
- flotante: float, double, long double
- Booleanos: bool Vacío: void

1992:

Todo programa debe tener la función principal main()

1990:

La palabra reservada void define en C++ el concepto de no existencia o no atribución de un tipo en una variable o declaración

1993:

Los objetos son abstraídos mediante una clase.

1997:

El sistema proporciona un constructor de copia, si no se define.

1998

Constructores + Memoria heap

1999:

Los destructores son funciones miembro especiales llamadas automáticamente en la ejecución del programa

# Versiones C++

**C++98/C++03:** Esta es la versión original y oficial del estándar ISO/IEC para C++. Fue publicada en 1998 y revisada en 2003. Introdujo muchas características fundamentales de C++, como clases, herencia, plantillas y excepciones.

---

**C++11:** También conocido como C++0x durante su desarrollo, esta versión fue publicada en 2011. Introdujo características significativas, como el soporte para programación concurrente con hilos, palabras clave auto y decltype, lambdas, bucles for basados en rangos y la biblioteca de punteros inteligentes.


# Versiones C++



**C++14:** Esta versión se publicó en 2014 y se centra principalmente en mejoras de rendimiento y facilidad de uso en comparación con C++11. Introduce características como funciones genéricas mejoradas, inicialización agregada, lambdas genéricas y literales binarios.

---

**C++17:** Publicado en 2017, C++17 incluye varias mejoras y extensiones notables. Introduce características como estructuración de declaración if, inicialización en sitio para estructuras, plantillas constexpr, if constexpr, capturas por valor en lambdas, entre otros.




# Versiones C++



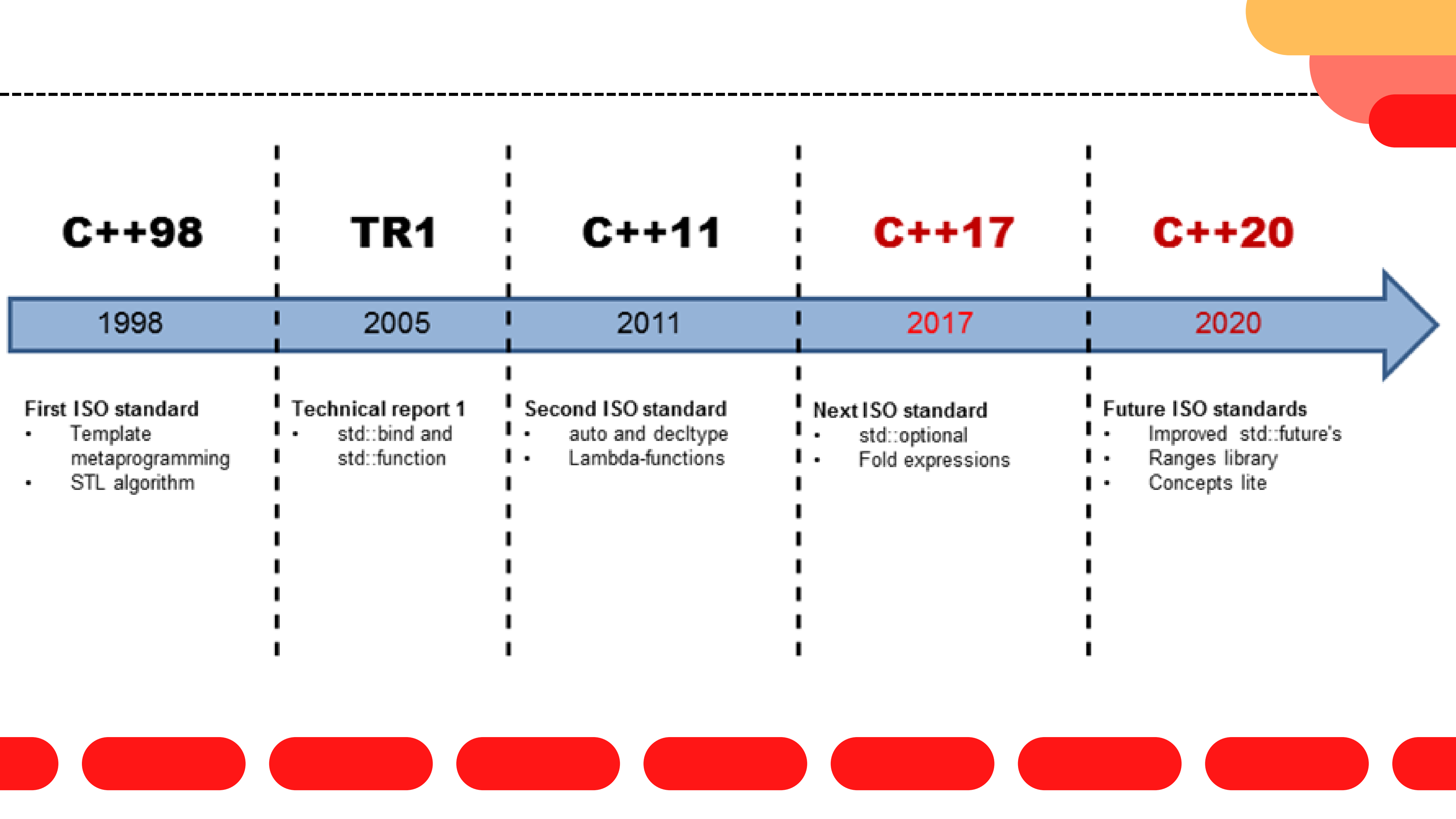
**C++20:** Esta versión se publicó en 2020 y presenta una amplia gama de nuevas características y mejoras. Algunas de las características destacadas incluyen módulos, conceptos, rangos, programación asíncrona y paralela, operadores de nave espacial [ $\leftarrow$ ,  $\rightarrow$ ], entre otros.

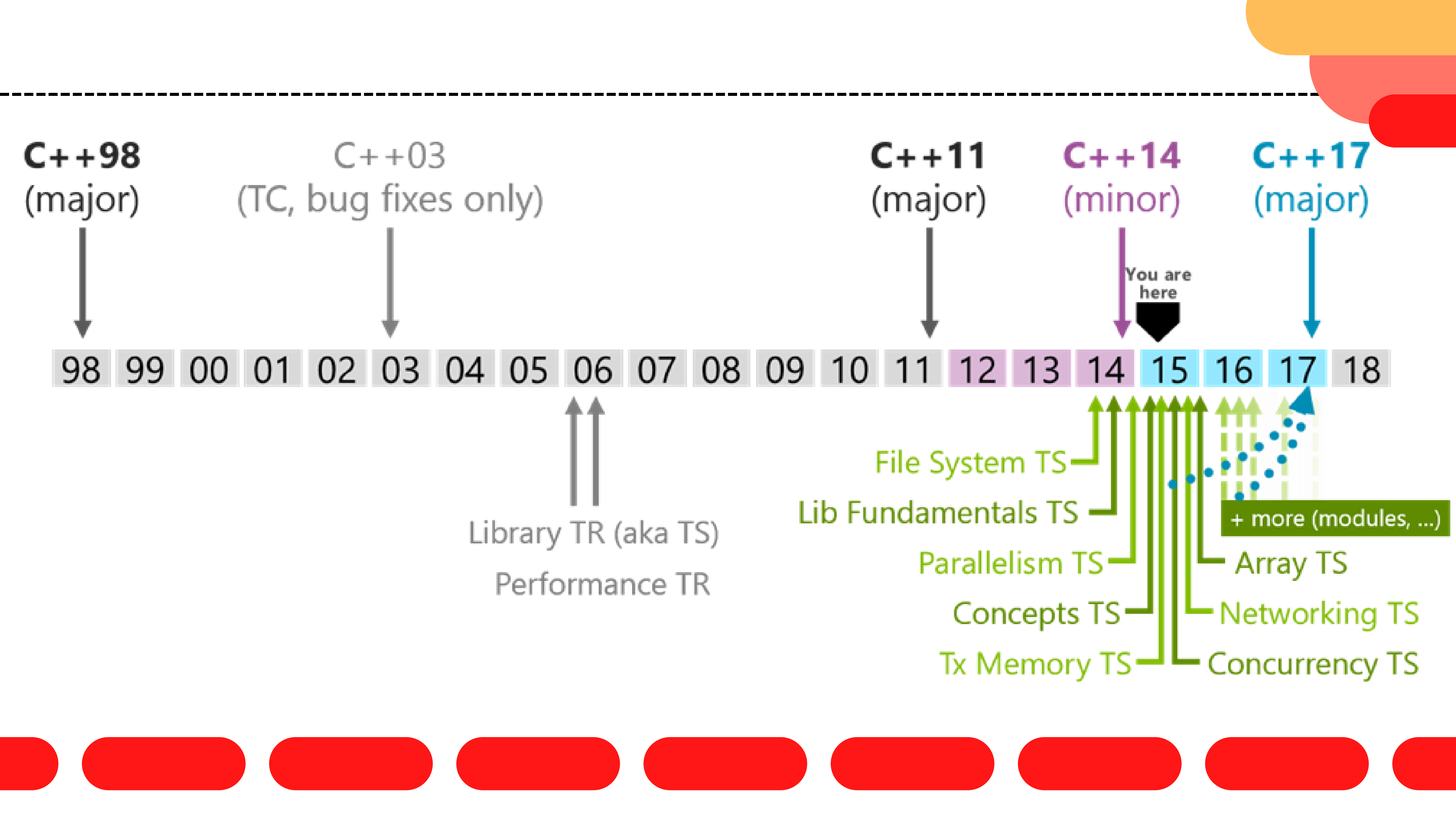
---

Es importante tener en cuenta que estas son solo algunas de las versiones principales de C++. A medida que el lenguaje evoluciona, se introducen nuevas características y mejoras.









# Ubicación en Ranking y Utilidad del lenguaje

C++

Ranking

En cuanto popularidad

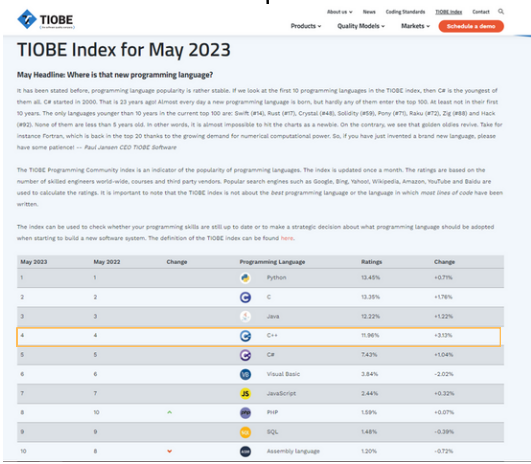
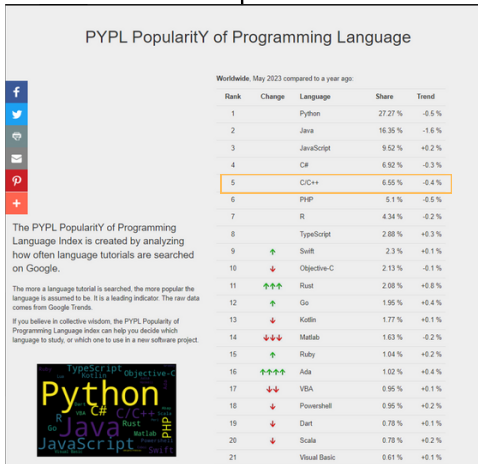
En cuanto desarrollo web

En cuanto desarrollo de sistemas operativos

En cuanto videojuegos

Según PYPL (Popularity of Programming Language)

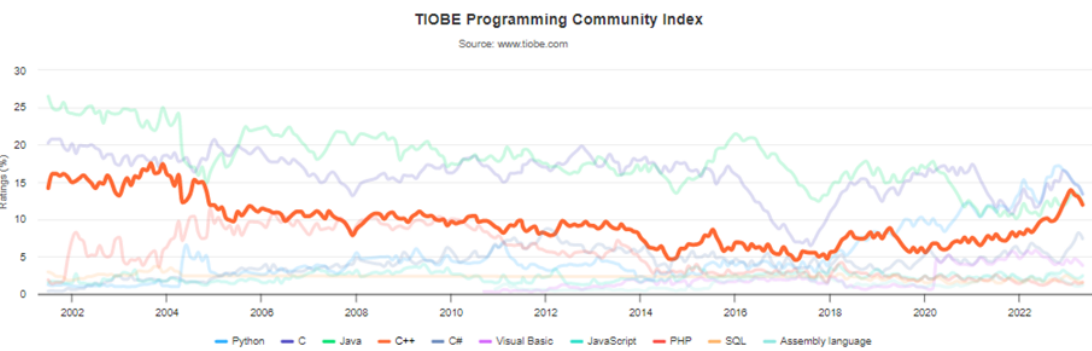
Según TIOBE (programming community index)



En términos de popularidad y uso en la industria, el lenguaje de programación C++ sigue siendo muy relevante y ampliamente utilizado en diversos campos.

C++ no es un lenguaje usual para el desarrollo de aplicativos webs. A pesar de que hay varios programas web hechos con este lenguaje, el apartado y funciones que se elaboran con lenguaje C no tiene opciones inteligentes.

C++ es un lenguaje de programación ampliamente utilizado en el desarrollo de sistemas operativos. A lo largo de los años, varios sistemas operativos importantes han sido escritos total o parcialmente en C++, incluyendo Windows, macOS, Linux y Android.



1-	C: C es conocido por su eficiencia y capacidad para realizar operaciones de bajo nivel, lo que lo convierte en una opción popular para el desarrollo de sistemas operativos.
2-	C++: C++ ofrece un rendimiento similar a C y también proporciona abstracciones de alto nivel que pueden facilitar el desarrollo de sistemas operativos complejos.
3-	Assembly: El ensamblador permite un control muy bajo nivel y un acceso directo al hardware, lo que es esencial para tareas como la administración de interrupciones y la programación de dispositivos específicos.

profile

Creamos ▾ Nuestra esencia Personas Innovación Blog ☰

### C++

Este lenguaje de programación es uno de los más utilizados en el sector por profesionales. Es un lenguaje popular en los títulos AAA, se utiliza en videojuegos para PlayStation y Xbox, y en juegos independientes. Se trata del lenguaje más compatible con la mayoría de los motores de juego y tiene un tiempo de ejecución bastante rápido. Por otro lado, permite a los desarrolladores tener un control amplio sobre el hardware, la gestión de la memoria y los gráficos, y aunque al principio puede resultar complejo de utilizar, una vez te haces a él, podrás manejar cualquier otro lenguaje.

### C Sharp

C# es un lenguaje de programación muy popular, sobre todo en entornos Windows. Es un poco menos flexible y compatible que C++, pero algunos motores como Unity permiten programar con él y no está limitado a un determinado sistema operativo o plataforma; se pueden crear juegos para iOS, Android, Windows Play Station y Xbox. Es un lenguaje más fácil de aprender que el C++ y será una buena opción si estás empezando o si quieres que lo disfruten más personas, con independencia del dispositivo.

### Java

Se trata de un lenguaje frecuentemente utilizado y presenta muchas similitudes con C++. Su principal característica es la versatilidad, ya que se puede utilizar en todas las plataformas, dispone de gran cantidad de frameworks para el desarrollo 3D, ofrece módulos de código abierto y su modelo se puede actualizar constantemente. ¿El problema? Que se ejecuta dentro de su máquina virtual, y esto supone una pérdida de rendimiento.

### JavaScript

Este es uno de los lenguajes más utilizados en el desarrollo de videojuegos web y de navegador. La mayoría de motores de videojuegos son compatibles con JavaScript, y cuenta con múltiples frameworks para 3D y una gran variedad de bibliotecas. Además, algunos motores de videojuegos como Unity lo utilizan, por lo que podremos usarlo para crear todo tipo de scripts dentro del juego.

### Python

A pesar de no ser un lenguaje de programación exclusivo para la creación de videojuegos, Python es un lenguaje muy flexible y potente para esto. Su ejecución es mucho más simple que la de otros lenguajes (permite plasmar ideas complejas con pocas líneas de código), y su framework Pygame permite a los desarrolladores crear prototipos de sus videojuegos de manera rápida y sencilla, y funciona prácticamente en todas las plataformas y sistemas operativos.

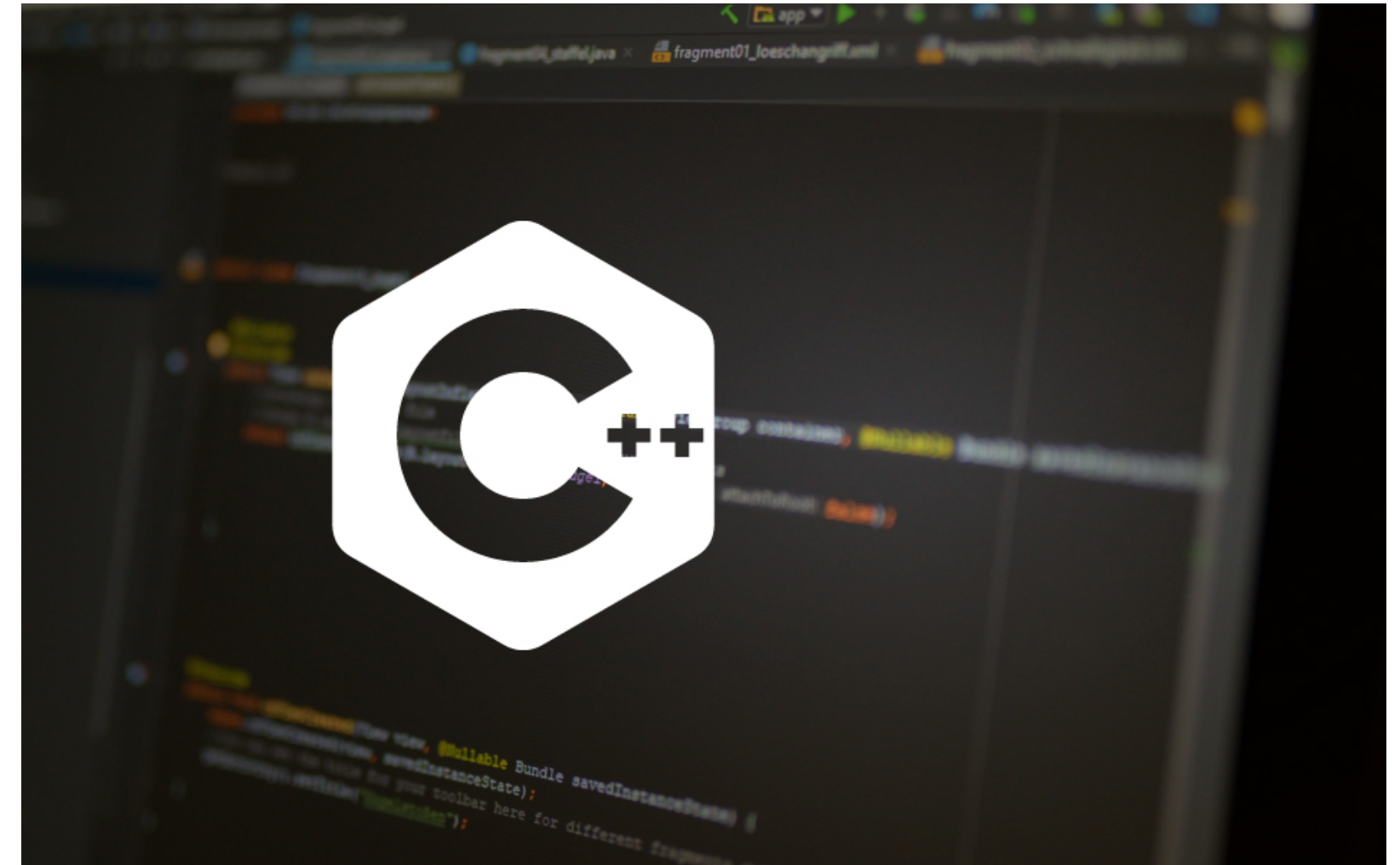
### Lua

Finalmente, Lua es un lenguaje de programación sencillo, rápido y fácil de aprender. Compatible con lenguajes más complejos y de rápida ejecución, también se usa para aplicaciones web y procesamiento de imágenes. Este lenguaje es especialmente útil para proyectos independientes y programadores que estén empezando en la profesión.

C++  
Es uno de los lenguajes de programación más utilizados por los profesionales del sector. Se utiliza para algunas de las principales plataformas de hardware como PlayStation y Xbox, siendo uno de los más compatibles con la mayor parte de motores de juego. Unido a un rápido tiempo de ejecución, C++ permite a los desarrolladores tener un amplio control sobre el hardware. Dominar este tipo de programación nos facilitará en gran parte la enseñanza de otros lenguajes

# Utilidad de C++

El lenguaje C++ es uno de los lenguajes de programación más populares y ampliamente utilizados en la industria del software. Tiene una amplia gama de aplicaciones y es conocido por su eficiencia y flexibilidad. Aquí tienes algunas de las utilidades principales del lenguaje C++:



# Ejemplos

**Desarrollo de software de sistemas: C++ se utiliza ampliamente para el desarrollo de sistemas operativos, controladores de dispositivos y otro software de nivel de sistema.**

**Desarrollo de aplicaciones de escritorio: C++ es utilizado para crear aplicaciones de escritorio, como editores de texto, suites de productividad, herramientas de diseño gráfico y aplicaciones de juegos.**

**1. Desarrollo de aplicaciones de juegos: Muchos motores de juegos, como Unity y Unreal Engine, utilizan C++ como lenguaje de programación principal.**

# Clases:

La sintaxis para declarar una clase es: "class" seguido del nombre de la clase y el cuerpo encerrado entre llaves, con el ';' al final.



# Objetos:

Para crear objetos, simplemente se define una clase que especifique su estructura y comportamiento.

Usa la siguiente sintaxis: `NombreClase nombreObjeto;`

El objeto puede llamar a los atributos y métodos de dicha clase con el operador de acceso `''`.



# Librerías:

`#include <iostream>`

Es parte de la librería estándar de C++ que permite la salida y entrada de datos.

`Cout<<`

Esto se usa para comunicarse con el usuario, le envía un mensaje (si decide escribir un mensaje) solicitando un dato.

`Cin>>`

Lee el dato que le manda el `cout<<` y se lo asigna a una variable.

`using namespace std;`

Es una instrucción que usa C++ para evitar repetir "std: :". Con este prefijo se accede a los elementos.



# Métodos:

Aquí se conocen como funciones.

Hay dos formas para implementar un método:

1. Se puede declarar el método y definir su cuerpo al mismo tiempo, dentro de la clase.

```
1 //Ejemplo
2 class MiClase {
3     private:
4         int numero;
5
6     public:
7         void establecerNumero(int n) {
8             numero = n;
9         }
10 };
```

# Métodos:

2. Se puede declarar el método dentro de la clase, especificando el tipo de retorno con sus parámetros y luego definir el cuerpo fuera de clase. Para esto, debe llamar a la clase donde se declaró acompañado del operador '::'.

```
1 //Ejemplo
2 class MiClase {
3     private:
4         int numero;
5
6     public:
7         void establecerNumero(int n);
8 };
9
10 void MiClase::establecerNumero(int n) {
11     numero = n;
12 }
```

# Métodos:

## Destructor:

Un destructor es una función miembro que destruye un objeto cuando ya no se lo necesita o cuando queda fuera de alcance, se ejecuta de manera automática. Esto para liberar recursos, cerrar archivos o simplemente liberar memoria.

Se identifica por `~` seguido del nombre de la clase, paréntesis y llaves.

## Composición:

Una clase contiene instancias de otras clases como miembros, estas son clases contenedoras.

## Agregación:

Aquí no hay dependencia entre clases, es decir si la clase contenedora es destruida, los elementos seguirán existiendo.

# Relaciones

```
#include <iostream>
#include <string>

using namespace std;

class Microprocesador {
private:
    string modelo;
public:
    Microprocesador(const string& modelo) : modelo(modelo) {}

    string getModelo() {
        return modelo;
    }
};

class USB {
private:
    string tipo;
public:
    USB(const string& tipo) : tipo(tipo) {}

    string getTipo() {
        return tipo;
    }
};

class Computador {
private:
    Microprocesador microprocesador;
    USB* usb;
public:
    Computador(const string& modeloMicroprocesador, const string& tipoUSB)
        : microprocesador(modeloMicroprocesador), usb(new USB(tipoUSB)) {}

    ~Computador() {
        delete usb;
    }

    string getModeloMicroprocesador() {
        return microprocesador.getModelo();
    }

    string getTipoUSB() {
        return usb->getTipo();
    }
};

int main() {
    Computador miComputador("Intel Core i7", "USB 3.0");

    cout << "Microprocesador: " << miComputador.getModeloMicroprocesador() << endl;
    cout << "Tipo de USB: " << miComputador.getTipoUSB() << endl;

    return 0;
}
```

# Herencia y Polimorfismo

El polimorfismo, es la habilidad de los objetos de diferentes clases que están relacionados mediante la herencia para responder en forma diferente al mismo mensaje (es decir, a la llamada de función miembro). El mismo mensaje que se envía a muchos tipos de objetos diferentes toma "muchas formas", y de ahí viene el término polimorfismo.

```
1  #include <iostream>
2
3  class Animal {
4  public:
5      virtual void sonido() {
6          |   std::cout << "Sonido desconocido" << std::endl;
7          |   }
8      };
9
10 class Perro : public Animal {
11 public:
12     void sonido() override {
13         |   std::cout << "Guau!" << std::endl;
14         |   }
15     };
16
17 class Gato : public Animal {
18 public:
19     void sonido() override {
20         |   std::cout << "Miau!" << std::endl;
21         |   }
22     };
23
24 int main() {
25     Animal* animal1 = new Perro();
26     Animal* animal2 = new Gato();
27     animal1->sonido(); // Imprime "Guau!"
28     animal2->sonido(); // Imprime "Miau!"
29     delete animal1;
30     delete animal2;
31     return 0;
32 }
```



# GUI



Las interfaces gráficas de usuario, también conocidas como GUI, son medios o programas que actúan como intermediarios para facilitar la comunicación y el uso efectivo del software por parte del usuario. Estas interfaces utilizan imágenes y otros elementos gráficos para representar información y acciones con las que el usuario puede interactuar de forma intuitiva.



# Desarrollo de las GUI en C++

En el desarrollo de interfaces en C++, existen diversas alternativas disponibles:

**Librerías y bibliotecas dedicadas.**

**Programación directa con uso de bibliotecas a bajo nivel.**

**Frameworks para el desarrollo del software.**

**Integración con otros lenguajes de scripting.**

# Qt y Frameworks del lenguaje.

Qt es un framework que se caracteriza por proporcionar un conjunto de componentes y herramientas para desarrollar e integrar interfaces de usuario para diversas aplicaciones. Qt ofrece su propio entorno de desarrollo visual, en donde es posible arrastrar los elementos para “dibujar” las pantallas, siendo compatible con múltiples plataformas, lenguajes y ofreciendo una amplia variedad de funciones.





Form Editor

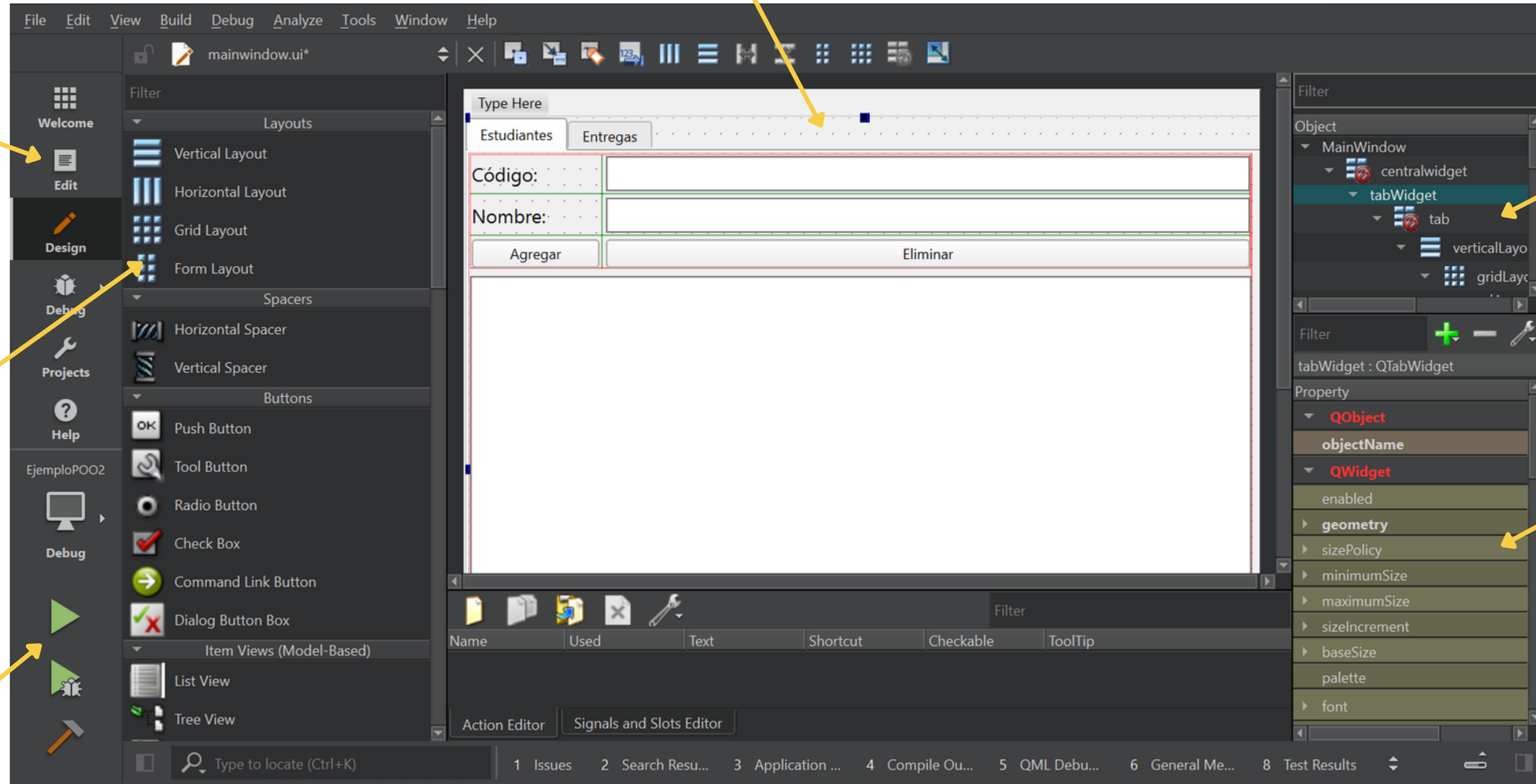
Editor

Controladores  
y Layouts

Depurador

Objetos

Propiedades



# Ejemplo de GUI en C++ con Qt:

Tomando como base el ejercicio del Quiz de Sistema Académico, hemos realizado un ejemplo con ciertas modificaciones en el entorno de desarrollo de Qt Creator, con la finalidad de gestionar los datos de las entregas de los alumnos en forma de tareas, quizes y parciales, tomando en cuenta los conceptos manejados a lo largo de nuestra presentación para poder agregar y eliminar los objetos.

