By 4ist4r

Nivel Básico

1. **Variables y tipos de datos**: En C++, se pueden declarar variables para almacenar valores. Los tipos de datos básicos son `int`, `char`, `float`, `double`, etc.
- Ejemplo: `int x = 5;`
1. **Operadores aritméticos**: Se pueden realizar operaciones aritméticas como suma, resta, multiplicación y división.
- Ejemplo: `int x = 5; int y = 3; int resultado = x + y;`
1. **Estructuras de control**: Se pueden utilizar estructuras de control como `if`, `else`, `switch`, `for`, `while`, etc.
- Ejemplo: `int x = 5; if (x > 10) { cout << "x es mayor que 10"; }`
1. **Funciones**: Se pueden definir funciones para reutilizar código.
- Ejemplo: `int suma(int x, int y) { return x + y; }`

Nivel Intermedio

1. **Arrays y vectores**: Se pueden utilizar arrays y vectores para almacenar colecciones de datos.
- Ejemplo: `int arr[5] = {1, 2, 3, 4, 5};`
1. **Punteros**: Se pueden utilizar punteros para acceder a la memoria de una variable.
- Ejemplo: `int x = 5; int* ptr = &x;`
1. **Clases y objetos**: Se pueden definir clases y objetos para encapsular datos y comportamiento.
- Ejemplo: `class Persona { public: string nombre; int edad; };`
1. **Herencia y polimorfismo**: Se pueden utilizar la herencia y el polimorfismo para crear jerarquías de clases y comportamientos.
- Ejemplo: `class Empleado : public Persona { public: int salario; };`

By 4ist4r

Nivel Avanzado

1. **Plantillas**: Se pueden utilizar plantillas para crear código genérico que puede trabajar con diferentes tipos de datos.
- Ejemplo: `template <typename T> T max(T a, T b) { return (a > b) ? a : b; }`
1. **Excepciones**: Se pueden utilizar excepciones para manejar errores y excepciones en el código.
- Ejemplo: `try { // código que puede lanzar una excepción } catch (exception& e) { // manejo de la excepción }`
1. **Programación concurrente**: Se pueden utilizar técnicas de programación concurrente para aprovechar al máximo los recursos del procesador.
- Ejemplo: `thread t1(funcion1); thread t2(funcion2);`
1. **Programación de bajo nivel**: Se pueden utilizar técnicas de programación de bajo nivel para acceder a la memoria y los recursos del sistema operativo.
- Ejemplo: `void* ptr = malloc(10);`

Nivel Experto

1. **Meta-programación**: Se pueden utilizar técnicas de meta-programación para crear código que puede generar código en tiempo de compilación.
- Ejemplo: `template <typename T> struct Meta { // código que genera código };`
1. **Optimización de código**: Se pueden utilizar técnicas de optimización de código para mejorar el rendimiento del programa.
- Ejemplo: `#pragma omp parallel for`
1. **Programación de sistemas**: Se pueden utilizar técnicas de programación de sistemas para crear programas que interactúan con el sistema operativo y los dispositivos periféricos.
- Ejemplo: `int fd = open("archivo.txt", O_RDONLY);`

1. **Seguridad**: Se pueden utilizar técnicas de seguridad para proteger el programa y los datos contra ataques y vulnerabilidades.
- Ejemplo: `#include <openssl/aes.h>`