

Introducción a la Programación C++

ÍNDICE

1	Introducción a C++.	2
1.1	Introducción a C++.	2
1.2	Elementos del lenguaje C++.	2
1.3	Estructura de un programa C++.	2
1.4	Bibliotecas C++.	3
1.5	Constantes, variables y tipos.	4
1.6	Expresiones básicas.	5
1.7	Operadores.	5
1.8	Entradas/Salidas.	6
2	Hola Mundo.	7

1 Introducción a C++.

1.1 Introducción a C++.

C++ es un lenguaje de programación de propósito general, que combina la programación orientada a objetos y la programación procedural. Es un lenguaje potente y ampliamente utilizado en una variedad de aplicaciones, desde sistemas embebidos hasta desarrollo de videojuegos y software de escritorio.

La principal diferencia radica en el enfoque de resolución de problemas: la programación procedural se centra en procedimientos y funciones, mientras que la programación orientada a objetos se centra en objetos y su interacción. La POO tiende a ser más modular, flexible y fácil de mantener a medida que los programas crecen en tamaño y complejidad.

1.2 Elementos del lenguaje C++.

Los elementos más simples que integran un programa escrito en C++ son las palabras, símbolos y las reglas para su formación.

- **Palabras Clave (Keywords):** Son palabras reservadas por el lenguaje que tienen un significado específico y no pueden ser utilizadas como nombres de variables o funciones. Algunas palabras clave en C++ son **int**, **float**, **if**, **else**, **for**, **while**, **class**, entre otras.
- **Identificadores (Identifiers):** Son nombres dados a entidades como variables, funciones, clases, etc. Los identificadores deben seguir ciertas reglas, como comenzar con una letra o un guión bajo, y pueden contener letras, dígitos y guiones bajos.
- **Comentarios (Comments):** Son texto ignorado por el compilador y se utilizan para documentar el código y hacerlo más legible. En C++, los comentarios de una sola línea comienzan con `//`, mientras que los comentarios de varias líneas están entre `/*` y `*/`.
- **Delimitadores:** Son símbolos que indican el comienzo o fin, como por ejemplo `%`, `[]`, `{}`, etc.

1.3 Estructura de un programa C++.

Un programa básico en C++ tiene la siguiente estructura:

```
cpp Copy code

#include <iostream> // Directiva de preprocesador

// Función principal
int main() {
    // Declaraciones de variables

    // Instrucciones

    return 0; // Fin del programa
}
```

- **#include <iostream>**: Esta línea incluye una **biblioteca estándar** de C++ que proporciona entrada y salida estándar.
- **int main() { ... }**: Esta es la función principal del programa donde comienza la ejecución. El tipo de retorno int indica que la función devuelve un entero.
- **// Declaraciones de variables**: Aquí se pueden declarar variables que se utilizarán en el programa.
- **// Instrucciones**: Aquí se escriben las instrucciones que se ejecutarán dentro de la función main().
- **return 0;**: Esta instrucción indica el final exitoso del programa.

1.4 Bibliotecas C++.

C++ ofrece una variedad de bibliotecas estándar que proporcionan funcionalidades esenciales para el desarrollo de aplicaciones en diferentes áreas. Aquí están algunas de las bibliotecas principales de C++:

Biblioteca Estándar de C++ (STL - Standard Template Library): Esta es una de las bibliotecas más fundamentales en C++. Proporciona contenedores de datos (como vectores, listas, conjuntos, mapas, etc.), algoritmos de manipulación de datos (como clasificación, búsqueda, transformación, etc.) y utilidades (como iteradores, funciones de comparación, etc.).

- **iostream**: Esta biblioteca proporciona funcionalidades para entrada y salida estándar en C++. Incluye objetos como std::cin para entrada desde el teclado y std::cout para salida a la consola.
- **cmath**: Esta biblioteca proporciona funciones matemáticas comunes como trigonométricas, exponenciales, logarítmicas, funciones de redondeo, etc.

- **string:** Esta biblioteca proporciona funcionalidades para la manipulación de cadenas de texto, como concatenación, búsqueda, extracción de substrings, etc.
- **algorithm:** Esta biblioteca proporciona una variedad de algoritmos estándar que operan en contenedores de datos, como ordenamiento, búsqueda, manipulación de rangos, entre otros.
- **vector:** Esta biblioteca proporciona una implementación de un contenedor de datos dinámico llamado vector, que es similar a un array dinámico en otros lenguajes de programación.
- **map y set:** Estas bibliotecas proporcionan implementaciones de estructuras de datos de asociación, como mapas (que son colecciones de pares clave-valor) y conjuntos (que son colecciones de valores únicos).
- **fstream:** Esta biblioteca proporciona funcionalidades para entrada y salida de archivos en C++, permitiendo la lectura y escritura de datos en archivos en el sistema de archivos del sistema operativo.

Estas son solo algunas de las bibliotecas principales en C++, pero hay muchas otras disponibles para una amplia gama de propósitos, incluyendo procesamiento de cadenas, manipulación de tiempo, programación de red, entre otros. Además de las bibliotecas estándar, también hay bibliotecas externas y de terceros que puedes usar para ampliar las capacidades de C++.

1.5 Constantes, variables y tipos.

- **Constantes:** Son valores que no cambian durante la ejecución del programa. Se pueden clasificar en constantes literales (como 5, 3.14, 'A') y constantes simbólicas (definidas con const).
- **Variables:** Son espacios de memoria reservados para almacenar datos que pueden cambiar durante la ejecución del programa. Se declaran especificando su tipo y nombre.
- **Tipos de Datos:** C++ tiene varios tipos de datos, incluyendo enteros (int, long, short), números en coma flotante (float, double), caracteres (char), booleanos (bool), entre otros. También permite la definición de tipos de datos personalizados mediante estructuras y clases.

1.6 Expresiones básicas.

- **Expresiones Aritméticas:** Combinación de constantes, variables y operadores aritméticos que producen un valor.

```
cpp

int resultado = 2 * (3 + 4) - 5; // Expresión aritmética
```

- **Expresiones de Asignación:** Se utiliza para asignar un valor a una variable.

```
cpp

int x = 10;
x += 5; // Equivalente a x = x + 5
```

- **Expresiones Relacionales y Lógicas:** Se utilizan para comparar valores y producir un resultado lógico (true o false).

```
cpp

int a = 5, b = 10;
bool esMayor = (a > b); // false
bool esIgual = (a == b); // false
```

1.7 Operadores.

Operadores Aritméticos:

- + (Suma): Suma dos operandos.
- - (Resta): Resta el segundo operando del primero.
- * (Multiplicación): Multiplica dos operandos.
- / (División): Divide el primer operando por el segundo.
- % (Módulo): Devuelve el resto de la división del primer operando por el segundo.

Operadores de Asignación:

- = (Asignación): Asigna el valor de la expresión del lado derecho al operando del lado izquierdo.
- +=, -=, *=, /=, %= (Operadores compuestos de asignación): Realizan una operación aritmética entre los operandos y asignan el resultado al operando de la izquierda.

Operadores de Comparación:

- == (Igualdad): Comprueba si dos operandos son iguales.
- != (Diferencia): Comprueba si dos operandos son diferentes.

- <, > (Menor que, Mayor que): Comprueban si el operando de la izquierda es menor o mayor que el operando de la derecha.
- <=, >= (Menor o igual que, Mayor o igual que): Comprueban si el operando de la izquierda es menor o igual o mayor o igual que el operando de la derecha.

Operadores Lógicos:

- && (Y lógico): Devuelve verdadero si ambos operandos son verdaderos.
- || (O lógico): Devuelve verdadero si al menos uno de los operandos es verdadero.
- ! (Negación lógica): Invierte el valor de verdad de su operando.

Operadores de Incremento/Decremento:

- ++ (Incremento): Aumenta en uno el valor de la variable.
- -- (Decremento): Disminuye en uno el valor de la variable.

Operadores de Ternario:

- ?: (Operador condicional ternario): Evalúa una expresión condicional y devuelve un valor dependiendo del resultado de la evaluación.

Operadores Bit a Bit:

- & (AND bit a bit), | (OR bit a bit), ^ (XOR bit a bit): Realizan operaciones lógicas bit a bit entre los operandos.
- << (Desplazamiento a la izquierda), >> (Desplazamiento a la derecha): Desplazan los bits de un operando hacia la izquierda o hacia la derecha.

Estos son algunos de los operadores más comunes en C++. Cada operador tiene su propia precedencia y asociatividad, lo que afecta el orden en que se evalúan las expresiones. Es importante comprender estos conceptos para escribir código claro y preciso en C++.

1.8 Entradas/Salidas.

- **Entrada de Datos:** Se utiliza para recibir datos del usuario.

```
cpp

int edad;
std::cout << "Introduce tu edad: ";
std::cin >> edad;
```

- **Salida de Datos:** Se utiliza para mostrar resultados al usuario.

cpp

```
int suma = 5 + 3;
std::cout << "La suma es: " << suma << std::endl;
```

En este ejemplo, `using namespace std;` permite que el programa utilice `cout` directamente sin tener que escribir `std::cout`. Sin embargo, ten en cuenta que algunas personas prefieren evitar `using namespace std;` para evitar posibles conflictos de nombres o para mantener un código más explícito sobre las dependencias del espacio de nombres. En su lugar, prefieren escribir `std::cout` para indicar claramente que están utilizando una función del espacio de nombres `std`.

cpp

```
#include <iostream>

using namespace std; // Permite usar cout sin std::

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

2 Hola Mundo.

Sigue los siguientes pasos:

- **Instala Visual Studio Code:** Si aún no tienes Visual Studio Code instalado en tu sistema, descárgalo e instálalo desde el sitio web oficial: [Descargar Visual Studio Code](#)
- **Instala el Compilador de C++:** Asegúrate de tener un compilador de C++ instalado en tu sistema. Cpp.
- **Abre Visual Studio Code:** Abre Visual Studio Code desde tu menú de aplicaciones o buscador de tu sistema.
- **Crea una Nueva Carpeta:** Crea una nueva carpeta en tu sistema donde desees guardar tu proyecto de C++.
- **Abre la Carpeta en Visual Studio Code:** En Visual Studio Code, abre la carpeta que acabas de crear seleccionando "File" (Archivo) -> "Open Folder" (Abrir Carpeta) y luego selecciona la carpeta que creaste.

- **Crea un Nuevo Archivo:** En Visual Studio Code, haz clic en el icono de archivo nuevo en la barra lateral izquierda o presiona Ctrl + N (o Cmd + N en macOS) para crear un nuevo archivo.
- **Escribe el Código "Hola Mundo":** En el archivo recién creado, escribe el siguiente código C++ para imprimir "Hola Mundo" en la consola:

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hola Mundo" << endl;
    return 0;
}
```

3 Sentencias de control.

Las **sentencias de control en C++** son herramientas fundamentales para dirigir el flujo de ejecución de un programa. Hay tres tipos principales de sentencias de control:

- **Sentencias de selección(if- switch):** Estas permiten tomar decisiones basadas en ciertas condiciones.
- **Sentencias de iteración (bucles) (while- do while- for):** Permiten ejecutar un bloque de código repetidamente mientras se cumple una condición.
- **Sentencias de salto:** Permiten transferir el control de ejecución a diferentes partes del programa.

1. Sentencias de selección:

- **if:** Se utiliza para ejecutar un bloque de código si se cumple una condición.

```
cpp Copy code
int edad = 20;
if (edad >= 18) {
    cout << "Eres mayor de edad";
}
```

- **if-else:** Se utiliza para ejecutar un bloque de código si se cumple una condición, y otro bloque si no se cumple.

```
cpp Copy code  
  
int num = 10;  
if (num % 2 == 0) {  
    cout << "El número es par";  
} else {  
    cout << "El número es impar";  
}
```

- **if-else if-else:** Se utiliza para encadenar múltiples condiciones.

```
cpp Copy code  
  
int score = 85;  
if (score >= 90) {  
    cout << "A";  
} else if (score >= 80) {  
    cout << "B";  
} else if (score >= 70) {  
    cout << "C";  
} else {  
    cout << "F";  
}
```

2. Sentencias de iteración (bucles):

- **while:** Se ejecuta un bloque de código mientras se cumpla una condición.

```
cpp Copy code  
  
int i = 1;  
while (i <= 5) {  
    cout << i << endl;  
    i++;  
}
```

- **do-while:** Similar a `while`, pero garantiza que el bloque de código se ejecute al menos una vez, ya que evalúa la condición después de ejecutar el bloque de código.

```
cpp Copy code  
  
int i = 1;  
do {  
    cout << i << endl;  
    i++;  
} while (i <= 5);
```

- **for:** Se utiliza para ejecutar un bloque de código un número específico de veces.

```
cpp Copy code  
  
for (int i = 1; i <= 5; i++) {  
    cout << i << endl;  
}
```

3. Sentencias de salto:

- **break:** Se utiliza para salir de un bucle.


```
cpp Copy code  
  
for (int i = 1; i <= 10; i++) {  
    if (i == 5) {  
        break; // Sale del bucle cuando i es igual a 5  
    }  
    cout << i << endl;  
}
```

- **continue:** Se utiliza para pasar a la siguiente iteración de un bucle.

```
cpp Copy code  
  
for (int i = 1; i <= 5; i++) {  
    if (i == 3) {  
        continue; // Salta la iteración cuando i es igual a 3  
    }  
    cout << i << endl;  
}
```

4 Implementación de un menú simple con las estructuras while – do y switch.

cpp

 Copy code

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

int main() {
    while (true) {
        cout << "Seleccione una opción:" << endl;
        cout << "1. Saludar" << endl;
        cout << "2. Despedirse" << endl;
        cout << "3. Salir del programa" << endl;
        cout << "Ingrese su opción: ";

        int opcion;
        cin >> opcion;

        switch (opcion) {
            case 1:
                cout << "¡Hola! ¡Bienvenido al programa!" << endl;
                break;
            case 2:
                cout << "¡Hasta luego! Que tengas un buen día." << endl;
                break;
            case 3:
                cout << "Saliendo del programa..." << endl;
                return 0; // Terminar el programa
            default:
                cout << "Opción inválida. Por favor, seleccione una opción válida." << endl;
        }
    }

    return 0;
}
```