Guía de C++ para Principiantes

Introducción

C++ es un lenguaje de programación poderoso y versátil que combina características de bajo y alto nivel. Desarrollado por Bjarne Stroustrup en 1985 como una extensión del lenguaje C, C++ es ampliamente utilizado en desarrollo de sistemas, videojuegos, aplicaciones de alto rendimiento y software embebido.

¿Por qué aprender C++?

- **Rendimiento**: C++ ofrece control directo sobre la memoria y recursos del sistema
- **Versatilidad**: Útil para sistemas operativos, videojuegos, aplicaciones web y más
- **Fundamento sólido**: Te ayuda a entender conceptos fundamentales de programación
- **Empleabilidad**: Muy demandado en la industria tecnológica

Configuración del Entorno

Compiladores Recomendados

- **GCC**: Disponible en Linux y Windows (MinGW)
- **Clang**: Multiplataforma con excelentes mensajes de error
- **MSVC**: Compilador de Microsoft para Windows

IDEs y Editores

- **Code::Blocks**: IDE gratuito multiplataforma
- **Dev-C++**: Simple y eficaz para principiantes
- **Visual Studio**: IDE profesional de Microsoft
- **VSCode**: Editor ligero con extensiones de C++

Compilación Básica

```cpp
// Guardar como hola.cpp
#include <iostream>

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

Compilar desde terminal:

```
g++ hola.cpp -o hola
./hola
```

Sintaxis Básica

Estructura de un Programa C++

```cpp
#include <iostream>  // Directiva de preprocesador
using namespace std; // Espacio de nombres

int main() {         // Función principal
    // Tu código aquí
    return 0;        // Valor de retorno
}
```

Variables y Tipos de Datos

Tipos Fundamentales

```cpp
// Números enteros
int edad = 25;
short año = 2024;
long poblacion = 1000000L;

// Números decimales
float precio = 19.99f;
double precision = 3.14159265359;

// Caracteres
char letra = 'A';
char nombre[] = "Juan";

// Booleanos
bool esVerdadero = true;
bool esFalso = false;
```

Declaración y Inicialización

```cpp
// Declaración simple
int numero;

// Inicialización
int cantidad = 10;

// Múltiples variables
int x = 5, y = 10, z = 15;

// Inicialización uniforme (C++11)
int valor{42};
```

Operadores

Operadores Aritméticos

```cpp
int a = 10, b = 3;
int suma = a + b;        // 13
int resta = a - b;       // 7
int producto = a * b;    // 30
int division = a / b;    // 3 (división entera)
int modulo = a % b;      // 1
```

Operadores de Comparación

```
bool igual = (a == b);        // false
bool diferente = (a != b);    // true
bool mayor = (a > b);         // true
bool menor = (a < b);         // false
bool mayorIgual = (a >= b);   // true
bool menorIgual = (a <= b);   // false
```

Operadores Lógicos

```
bool p = true, q = false;
bool and_logico = p && q;     // false
bool or_logico = p || q;      // true
bool not_logico = !p;         // false
```

Estructuras de Control

Condicionales

if, else if, else

```
int nota = 85;

if (nota >= 90) {
    cout << "Excelente" << endl;
} else if (nota >= 80) {
    cout << "Muy bueno" << endl;
} else if (nota >= 70) {
    cout << "Bueno" << endl;
} else {
    cout << "Necesita mejorar" << endl;
}
```

switch

```
char opcion = 'B';

switch (opcion) {
    case 'A':
        cout << "Opción A seleccionada" << endl;
        break;
    case 'B':
        cout << "Opción B seleccionada" << endl;
        break;
    case 'C':
        cout << "Opción C seleccionada" << endl;
        break;
    default:
        cout << "Opción no válida" << endl;
}
```

Bucles (Loops)

for

```
// Bucle for tradicional
for (int i = 1; i <= 10; i++) {
    cout << "Número: " << i << endl;
}

// Bucle for con rango (C++11)
int numeros[] = {1, 2, 3, 4, 5};
for (int num : numeros) {
    cout << num << " ";
}
```

while

```
int contador = 1;
while (contador <= 5) {
    cout << "Iteración: " << contador << endl;
    contador++;
}
```

do-while

```
int numero;
do {
    cout << "Ingresa un número positivo: ";
    cin >> numero;
} while (numero <= 0);
```

Funciones

Definición y Llamada

```
// Declaración de función
int sumar(int a, int b);

int main() {
    int resultado = sumar(5, 3);
    cout << "Resultado: " << resultado << endl;
    return 0;
}

// Definición de función
int sumar(int a, int b) {
    return a + b;
}
```

Tipos de Funciones

```cpp
// Función que no devuelve nada
void saludar() {
    cout << "¡Hola!" << endl;
}

// Función con parámetros por defecto
int potencia(int base, int exponente = 2) {
    int resultado = 1;
    for (int i = 0; i < exponente; i++) {
        resultado *= base;
    }
    return resultado;
}

// Función sobrecargada
int maximo(int a, int b) {
    return (a > b) ? a : b;
}

double maximo(double a, double b) {
    return (a > b) ? a : b;
}
```

Arrays y Strings

Arrays

```cpp
// Declaración e inicialización
int numeros[5] = {10, 20, 30, 40, 50};

// Acceso a elementos
cout << "Primer elemento: " << numeros[0] << endl;
numeros[2] = 35; // Modificar elemento

// Recorrer array
for (int i = 0; i < 5; i++) {
    cout << numeros[i] << " ";
}
```

Arrays Multidimensionales

```cpp
// Matriz 3x3
int matriz[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};

// Acceso a elementos
cout << matriz[1][2] << endl; // Imprime 6
```

Strings

```
#include <string>

// Strings de C (arrays de caracteres)
char nombre[] = "Carlos";
char apellido[20];
strcpy(apellido, "García");

// Strings de C++ (clase string)
string mensaje = "Hola mundo";
string saludo = "Buenos días";

// Operaciones con strings
string completo = nombre + " " + apellido;
cout << "Longitud: " << mensaje.length() << endl;
cout << "Carácter en posición 2: " << mensaje[2] << endl;
```

Punteros Básicos

Concepto de Punteros

```
int numero = 42;
int* puntero = &numero;  // Puntero apunta a la dirección de numero

cout << "Valor de numero: " << numero << endl;
cout << "Dirección de numero: " << &numero << endl;
cout << "Valor del puntero: " << puntero << endl;
cout << "Valor apuntado por puntero: " << *puntero << endl;
```

Operaciones con Punteros

```
void intercambiar(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int x = 10, y = 20;
    cout << "Antes: x = " << x << ", y = " << y << endl;

    intercambiar(&x, &y);

    cout << "Después: x = " << x << ", y = " << y << endl;
    return 0;
}
```

Clases y Objetos Básicos

Definición de una Clase

```
class Persona {
private:
    string nombre;
    int edad;

public:
    // Constructor
    Persona(string nom, int ed) {
        nombre = nom;
        edad = ed;
    }

    // Métodos
    void presentarse() {
        cout << "Hola, soy " << nombre << " y tengo " << edad << " años." << endl;
    }

    // Getters
    string getNombre() const {
        return nombre;
    }

    int getEdad() const {
        return edad;
    }

    // Setters
    void setEdad(int nuevaEdad) {
        if (nuevaEdad >= 0) {
            edad = nuevaEdad;
        }
    }
};
```

Uso de la Clase

```
int main() {
    // Crear objetos
    Persona persona1("Ana", 25);
    Persona persona2("Luis", 30);

    // Llamar métodos
    persona1.presentarse();
    persona2.presentarse();

    // Usar getters y setters
    cout << persona1.getNombre() << " tiene " << persona1.getEdad() << " años." << endl
;
    persona1.setEdad(26);
    cout << "Ahora " << persona1.getNombre() << " tiene " << persona1.getEdad() << "
años." << endl;

    return 0;
}
```

Ejemplos Prácticos

Ejemplo 1: Calculadora Simple

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

class Calculadora {
public:
    double sumar(double a, double b) { return a + b; }
    double restar(double a, double b) { return a - b; }
    double multiplicar(double a, double b) { return a * b; }
    double dividir(double a, double b) {
        if (b != 0) return a / b;
        else {
            cout << "Error: División por cero" << endl;
            return 0;
        }
    }
};

int main() {
    Calculadora calc;
    double num1, num2;
    char operador;

    cout << "Ingresa la operación (ej: 5 + 3): ";
    cin >> num1 >> operador >> num2;

    switch (operador) {
        case '+':
            cout << "Resultado: " << calc.sumar(num1, num2) << endl;
            break;
        case '-':
            cout << "Resultado: " << calc.restar(num1, num2) << endl;
            break;
        case '*':
            cout << "Resultado: " << calc.multiplicar(num1, num2) << endl;
            break;
        case '/':
            cout << "Resultado: " << calc.dividir(num1, num2) << endl;
            break;
        default:
            cout << "Operador no válido" << endl;
    }

    return 0;
}
```

Ejemplo 2: Sistema de Gestión de Estudiantes

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;

class Estudiante {
private:
    string nombre;
    int id;
    vector<double> calificaciones;

public:
    Estudiante(string nom, int identificacion) : nombre(nom), id(identificacion) {}

    void agregarCalificacion(double calificacion) {
        if (calificacion >= 0 && calificacion <= 100) {
            calificaciones.push_back(calificacion);
        }
    }

    double promedioCalificaciones() const {
        if (calificaciones.empty()) return 0;

        double suma = 0;
        for (double cal : calificaciones) {
            suma += cal;
        }
        return suma / calificaciones.size();
    }

    void mostrarInfo() const {
        cout << "Estudiante: " << nombre << " (ID: " << id << ")" << endl;
        cout << "Promedio: " << promedioCalificaciones() << endl;
        cout << "Calificaciones: ";
        for (double cal : calificaciones) {
            cout << cal << " ";
        }
        cout << endl << endl;
    }

    string getNombre() const { return nombre; }
    int getId() const { return id; }
};

int main() {
    vector<Estudiante> estudiantes;

    // Crear estudiantes
    estudiantes.push_back(Estudiante("María García", 001));
    estudiantes.push_back(Estudiante("Juan Pérez", 002));
    estudiantes.push_back(Estudiante("Ana López", 003));

    // Agregar calificaciones
    estudiantes[0].agregarCalificacion(85);
    estudiantes[0].agregarCalificacion(92);
    estudiantes[0].agregarCalificacion(78);

    estudiantes[1].agregarCalificacion(90);
    estudiantes[1].agregarCalificacion(88);
    estudiantes[1].agregarCalificacion(95);

    estudiantes[2].agregarCalificacion(76);
```

```
    estudiantes[2].agregarCalificacion(82);
    estudiantes[2].agregarCalificacion(89);

    // Mostrar información de todos los estudiantes
    cout << "=== REPORTE DE ESTUDIANTES ===" << endl;
    for (const auto& estudiante : estudiantes) {
        estudiante.mostrarInfo();
    }

    return 0;
}
```

Consejos para Continuar Aprendiendo

1. **Practica regularmente**: La programación se aprende haciendo
2. **Lee código de otros**: Estudia proyectos open source
3. **Construye proyectos**: Aplica lo aprendido en proyectos reales
4. **Aprende las mejores prácticas**: Código limpio y legible
5. **Explora temas avanzados**: STL, templates, programación orientada a objetos avanzada

Recursos Adicionales

- **Documentación oficial**: https://cplusplus.com/
- **Compilador online**: https://onlinegdb.com/
- **Ejercicios**: HackerRank, CodeForces, LeetCode
- **Libros recomendados**: “C++ Primer” de Stanley Lippman

¡Bienvenido al mundo de la programación en C++!