**Introduction to C++**

C++ là một ngôn ngữ lập trình đa năng, hiệu suất cao. Nó được phát triển bởi Bjarne Stroustrup tại Bell Labs từ năm 1979. C++ là phần mở rộng của ngôn ngữ lập trình C, bổ sung các tính năng như lớp, đối tượng và ngoại lệ.

**Basics of C++ Programming**

Sau đây là một số thành phần và khái niệm cơ bản trong lập trình C++:

**Including Libraries**

Trong C++, chúng ta sử dụng chỉ thị **#include** để đưa các thư viện hoặc tệp tiêu đề vào chương trình. Ví dụ: để đưa thư viện đầu vào/đầu ra chuẩn vào, chúng ta viết:

#include <iostream>

**Main Funtion**

Điểm vào của một chương trình C++ là hàm chính. Mọi chương trình C++ đều phải có một hàm chính:

int main() {

// Your code goes here

return 0;

}

**Input/Output**

Để thực hiện các thao tác nhập và xuất trong C++, chúng ta có thể sử dụng các đối tượng tích hợp std::cin cho đầu vào và std::cout cho đầu ra, có sẵn trong thư viện iostream. Dưới đây là ví dụ về cách đọc một số nguyên và in giá trị của nó:

#include <iostream>

int main() {

int number;

std::cout << "Enter an integer: ";

std::cin >> number;

std::cout << "You entered: " << number << '\n';

return 0;

}

**Variables and Data Types**

C++ có một số kiểu dữ liệu cơ bản để biểu diễn các giá trị số nguyên, số thực và ký tự:

int: giá trị số nguyên

float: giá trị số thực độ chính xác đơn

double: giá trị số thực độ chính xác kép

char: ký tự đơn

bool: giá trị boolean

Biến phải được khai báo với một kiểu dữ liệu trước khi có thể sử dụng:

int x;

float y;

double z;

char c;

bool b;

**Control Structures**

C++ cung cấp các cấu trúc điều khiển để thực thi có điều kiện và lặp lại, chẳng hạn như các câu lệnh if, else, while, for và switch.

**If-Else Statement**

if (condition) {

// Code to execute if the condition is true

} else {

// Code to execute if the condition is false

}

**While Loop**

while (condition) {

// Code to execute while the condition is true

}

**For Loop**

for (initialization; condition; update) {

// Code to execute while the condition is true

}

**Switch Statement**

switch (variable) {

case value1:

// Code to execute if variable == value1

break;

case value2:

// Code to execute if variable == value2

break;

// More cases...

default:

// Code to execute if variable does not match any case value

}

**Functions**

Hàm là các khối mã có thể tái sử dụng, có thể được gọi bằng các đối số để thực hiện một tác vụ cụ thể. Hàm được định nghĩa với kiểu trả về, tên, danh sách tham số và phần thân.

ReturnType functionName(ParameterType1 parameter1, ParameterType2 parameter2) {

// Function body

// ...

return returnValue;

}

Ví dụ, đây là một hàm cộng hai số nguyên và trả về kết quả:

int add(int a, int b) {

return a + b;

}

int main() {

int result = add(3, 4);

std::cout << "3 + 4 = " << result << '\n';

return 0;

}

Phần giới thiệu cơ bản về C++ này sẽ cung cấp cho bạn nền tảng vững chắc để học sâu hơn. Khám phá thêm các chủ đề như lớp, đối tượng, kế thừa, đa hình, mẫu và Thư viện Mẫu Chuẩn (STL) để hiểu sâu hơn về C++ và bắt đầu viết các chương trình nâng cao hơn.

Visit the following resources to learn more:

Free Resources

* [articleLearnC++](https://www.learncpp.com/" \t "https://roadmap.sh/_blank) : https://www.learncpp.com/
* [videoC++ Full Course by freeCodeCamp](https://youtu.be/vLnPwxZdW4Y" \t "https://roadmap.sh/_blank) : https://youtu.be/vLnPwxZdW4Y?si=THeT78czJDvtSbYF
* [videoC++ C++ Programming Course - Beginner to Advanced](https://www.youtube.com/watch?v=8jLOx1hD3_o" \t "https://roadmap.sh/_blank): https://youtu.be/8jLOx1hD3\_o?si=0ERLOFMl74sSnedN

**WHAT IS C++?**

C++ là một ngôn ngữ lập trình đa năng được Bjarne Stroustrup tạo ra như một phần mở rộng của ngôn ngữ lập trình C. Ngôn ngữ này được giới thiệu lần đầu tiên vào năm 1985 và cung cấp các tính năng hướng đối tượng như lớp và kế thừa. C++ được sử dụng rộng rãi trong nhiều ứng dụng như phát triển trò chơi, lập trình hệ thống, hệ thống nhúng và điện toán hiệu năng cao.

C++ là một ngôn ngữ kiểu tĩnh, nghĩa là kiểu của biến được xác định trong quá trình biên dịch và có một thư viện mở rộng gọi là Thư viện Chuẩn C++, cung cấp một tập hợp phong phú các hàm, thuật toán và cấu trúc dữ liệu cho nhiều tác vụ khác nhau.

C++ được xây dựng dựa trên các tính năng của C, do đó, hầu hết các chương trình C đều có thể được biên dịch và chạy bằng trình biên dịch C++

Ví dụ mã

Dưới đây là một ví dụ đơn giản về một chương trình C++ thể hiện một số tính năng thiết yếu của ngôn ngữ:

#include <iostream>

// A simple function to add two numbers

int add(int a, int b) {

return a + b;

}

class Calculator {

public:

// A member function to multiply two numbers

int multiply(int a, int b) {

return a \* b;

}

};

int main() {

int x = 5;

int y = 3;

// Using the standalone function 'add'

int sum = add(x, y);

std::cout << "Sum: " << sum << '\n';

// Using a class and member function

Calculator calc;

int product = calc.multiply(x, y);

std::cout << "Product: " << product << '\n';

return 0;

}

Trong chương trình trên, chúng ta định nghĩa một hàm đơn giản là add và một lớp Calculator với hàm thành viên là multiply. Hàm main minh họa cách sử dụng chúng để thực hiện các phép tính số học cơ bản. Truy cập các tài nguyên sau để tìm hiểu thêm:

**Why C++**

C++ là một ngôn ngữ lập trình phổ biến và được sử dụng rộng rãi vì nhiều lý do. Dưới đây là một số lý do tại sao bạn có thể chọn sử dụng C++:

***Performance(hiệu suất)***

C++ được thiết kế để mang lại hiệu suất và hiệu quả cao. Nó cung cấp khả năng kiểm soát chi tiết đối với tài nguyên hệ thống, giúp tối ưu hóa phần mềm của bạn dễ dàng hơn.

***Portability (tính khả chuyển)***

C++ được hỗ trợ trên nhiều kiến trúc máy tính và hệ điều hành khác nhau, cho phép bạn viết mã di động chạy trên nhiều nền tảng khác nhau mà không cần thực hiện các sửa đổi lớn

***Object-Oriented Programming***

C++ hỗ trợ lập trình hướng đối tượng (OOP) - một mô hình cho phép bạn thiết kế chương trình bằng cách sử dụng các lớp và đối tượng, dẫn đến khả năng tổ chức mã và tái sử dụng tốt hơn.

class MyClass {

public:

void myFunction() {

// Code here

}

};

int main() {

MyClass obj;

obj.myFunction();

}

***Support for low-level and high-level programming***

C++ cho phép bạn viết cả mã cấp thấp, như thao tác bộ nhớ, cũng như mã trừu tượng cấp cao, như tạo lớp và sử dụng Thư viện mẫu chuẩn (STL).

#include <iostream>

#include <vector>

int main() {

// Low-level programming

int number = 42;

int\* ptr\_number = &number;

// High-level programming

std::vector<int> myVector = {1, 2, 3};

for (const auto &i: myVector) {

std::cout << i << '\n';

}

}

***Extensive Libraries (thư viện mở rộng)***

C++ cung cấp nhiều thư viện và công cụ, chẳng hạn như Thư viện mẫu chuẩn (STL), Boost và Qt, cùng nhiều công cụ khác, có thể hỗ trợ phát triển dự án của bạn và giúp dự án hiệu quả hơn.

***Combination with C language (Kết hợp với ngôn ngữ C):***

C++ có thể được kết hợp với C, mang lại khả năng của cả hai ngôn ngữ và cho phép bạn tái sử dụng mã C hiện có. Bằng cách kết hợp các tính năng của C++, bạn có thể cải thiện mã và nâng cao chức năng của nó.

***Active Community (cộng đồng hoạt động)***

C++ đã tồn tại từ lâu và sở hữu một cộng đồng người dùng đông đảo, năng động, đóng góp vào sự phát triển của ngôn ngữ, thể hiện những ý tưởng mới và tham gia vào các cuộc thảo luận giúp ngôn ngữ này phát triển hơn nữa. Điều này giúp việc tìm kiếm giải pháp cho bất kỳ vấn đề nào bạn gặp phải trở nên dễ dàng hơn rất nhiều. Tóm lại, C++ mang lại sự cân bằng tuyệt vời giữa hiệu suất, tính di động và bộ tính năng, khiến nó trở thành một ngôn ngữ lập trình linh hoạt và mạnh mẽ, phù hợp với nhiều ứng dụng. Với các thư viện phong phú, cộng đồng năng động và sự phát triển liên tục, C++ là một lựa chọn tuyệt vời cho bất kỳ dự án phát triển phần mềm nào.

**C vs C++**

C và C++ là hai ngôn ngữ lập trình phổ biến với một số điểm tương đồng, nhưng cũng có những điểm khác biệt chính. C++ là phần mở rộng của ngôn ngữ lập trình C, với các tính năng bổ sung như lập trình hướng đối tượng, lớp và xử lý ngoại lệ. Mặc dù cả hai ngôn ngữ đều được sử dụng cho các tác vụ tương tự, nhưng chúng có cú pháp và ngữ nghĩa riêng, khiến chúng khác biệt với nhau.

***Syntax and Semantics (Cú pháp và ngữ nghĩa):***

**C**

C là ngôn ngữ lập trình thủ tục.

Tập trung vào hàm và lập trình có cấu trúc.

Không hỗ trợ đối tượng hoặc lớp.

Quản lý bộ nhớ được thực hiện thủ công, sử dụng các hàm như malloc và free.

#include<stdio.h>

void printHello() { printf("Hello, World!\n"); }

int main() { printHello(); return 0; }

**C++**

C++ vừa mang tính thủ tục vừa hướng đối tượng.

Hỗ trợ cả hàm và lớp.

Kết hợp nhiều mô hình lập trình khác nhau.

Quản lý bộ nhớ có thể được thực hiện thủ công (như C) hoặc dựa vào hàm tạo/hàm hủy và con trỏ thông minh.

#include<iostream>

class HelloWorld { public: void printHello() { std::cout << "Xin chào, Thế giới!\n"; } };

int main() { HelloWorld obj; obj.printHello(); return 0; }

***Code Reusability and Modularity (Khả năng tái sử dụng và tính mô-đun của mã)***

**C**

Khả năng tái sử dụng mã C đạt được thông qua các hàm và lập trình mô-đun.

Tính gắn kết cao và độ liên kết thấp đạt được thông qua thiết kế có cấu trúc.

Thư viện hàm có thể được tạo và tích hợp thông qua các tiêu đề.

**C++**

C++ cung cấp khả năng tái sử dụng mã tốt hơn với các lớp, tính kế thừa và tính đa hình.

Tính mô-đun của mã được cải thiện thông qua các không gian tên và hệ thống phân cấp hướng đối tượng được thiết kế tốt.

***Error Handling(Xử lý lỗi)***

**C**

Xử lý lỗi trong C chủ yếu được thực hiện thông qua mã trả về.

Không hỗ trợ ngoại lệ hoặc bất kỳ cơ chế xử lý lỗi tích hợp nào.

**C++**

Cung cấp khả năng xử lý ngoại lệ, có thể được sử dụng để xử lý các lỗi có thể xảy ra trong quá trình thực thi chương trình.

Cho phép bắt và xử lý ngoại lệ bằng các từ khóa try, catch và throw, giúp kiểm soát tốt hơn việc xử lý lỗi.

***Conclusion (Kết luận)***

Cả C và C++ đều là những ngôn ngữ mạnh mẽ với các tính năng và khả năng độc đáo. Trong khi C đơn giản hơn và tập trung vào lập trình thủ tục, C++ mang đến sự linh hoạt trong việc sử dụng các mô hình lập trình khác nhau và tổ chức mã được cải thiện. Hiểu được sự khác biệt giữa hai ngôn ngữ này có thể giúp bạn quyết định ngôn ngữ nào phù hợp hơn với nhu cầu và phong cách lập trình cụ thể của mình.