(Ghi chú)

Ngôn ngữ lập trình cpp

**Thực hiện: Nguyễn Văn Hào**

**Email:** [hao2205tb@gmail.com](mailto:hao2205tb@gmail.com)

(Tài liệu được biên xoạn lại từ nhiều nguồn và không dùng trong mục đích kinh doanh kiếm tiền)

# Mục lục

[Mục lục 1](#_Toc126691932)

[Tổng quan C++ 6](#_Toc126691933)

[C++ là ngôn ngữ lập trình mở rộng của C 6](#_Toc126691934)

[C++ là ngôn ngữ lập trình hướng đối tượng 6](#_Toc126691935)

[Có nhiều loại môi trường phát triển C++ khác nhau 7](#_Toc126691936)

[Các phiên bản của ngôn ngữ C++ 8](#_Toc126691937)

[C++ kế thừa ưu điểm về tốc độ xử lý của C 9](#_Toc126691938)

[Hiểu về C++ sẽ hiểu về máy tính 10](#_Toc126691939)

[Khả năng áp dụng vào các ngôn ngữ lập trình khác 10](#_Toc126691940)

[Ứng dụng và trò chơi trên điện thoại thông minh 11](#_Toc126691941)

[Phát triển hệ thống kinh doanh 11](#_Toc126691942)

[Ứng dụng WEB 11](#_Toc126691943)

[Một số dịch vụ được phát triển bằng C++ 11](#_Toc126691944)

[Chrome 12](#_Toc126691945)

[Cocos2d-x 12](#_Toc126691946)

[Unreal-Engine 13](#_Toc126691947)

[Hello world! 13](#_Toc126691948)

[Cấu trúc cơ bản của chương trình C++ 13](#_Toc126691949)

[Quy tắc viết chương trình C++ 14](#_Toc126691950)

[Hello world! 14](#_Toc126691951)

[Comment 14](#_Toc126691952)

[Nhập xuất 15](#_Toc126691953)

[4 lệnh nhập xuất cơ bản trong C++ 15](#_Toc126691954)

[Toán tử nhập xuất 15](#_Toc126691955)

[Cout 15](#_Toc126691956)

[Cerr 16](#_Toc126691957)

[Clog 16](#_Toc126691958)

[Cin 18](#_Toc126691959)

[Cin.ignore 18](#_Toc126691960)

[Cin.getline 20](#_Toc126691961)

[Toán tử 21](#_Toc126691962)

[Bảng toán tử so sánh 21](#_Toc126691963)

[Bảng toán tử logic 21](#_Toc126691964)

[Thứ tự ưu tiên toán tử logic trong C++ 21](#_Toc126691965)

[Bảng toán tử số học 22](#_Toc126691966)

[Toán tử tăng và giảm 22](#_Toc126691967)

[Sự Khác Biệt Giữa ++ i Và i ++ 22](#_Toc126691968)

[Toán tử 3 ngôi 23](#_Toc126691969)

[Bảng toán tử gán 24](#_Toc126691970)

[Phép gán đồng thời 24](#_Toc126691971)

[Kí tự 25](#_Toc126691972)

[Mã ký tự 25](#_Toc126691973)

[Lấy mã ASSCII 25](#_Toc126691974)

[Chuyển từ mã ascii sang ký tự 25](#_Toc126691975)

[Biểu diễn ký tự 26](#_Toc126691976)

[Chuỗi ký tự 26](#_Toc126691977)

[Ký tự đặc biệt 26](#_Toc126691978)

[Số 27](#_Toc126691979)

[Số nguyên 27](#_Toc126691980)

[Số thực 27](#_Toc126691981)

[Hằng số 27](#_Toc126691982)

[Kiểm tra số chẵn lẻ 27](#_Toc126691983)

[Cách 1: chia cho 2 lấy dư. 27](#_Toc126691984)

[Cách 2: and với 1 28](#_Toc126691985)

[Kiểm tra số nguyên 28](#_Toc126691986)

[Kiểm tra số nguyên tố 29](#_Toc126691987)

[Kiểm tra số âm, số dương 29](#_Toc126691988)

[Kiểm tra số chính phương 30](#_Toc126691989)

[Kiểm tra số hoàn hảo 31](#_Toc126691990)

[Tìm số đảo ngược 31](#_Toc126691991)

[Kiểm tra số đối xứng 32](#_Toc126691992)

[Số pi 33](#_Toc126691993)

[Sử dụng hằng số M\_PI trong cmath 33](#_Toc126691994)

[Sử dụng hằng số std::numbers::pi trong numbers 33](#_Toc126691995)

[Tự tạo số pi 33](#_Toc126691996)

[Biến 34](#_Toc126691997)

[Khai báo 34](#_Toc126691998)

[Khởi tạo 34](#_Toc126691999)

[Lệnh điều kiện 34](#_Toc126692000)

[If 34](#_Toc126692001)

[If đơn 34](#_Toc126692002)

[if else 35](#_Toc126692003)

[if elif else 35](#_Toc126692004)

[Switch case 36](#_Toc126692005)

[Cú pháp 36](#_Toc126692006)

[switch case trong C++ với kiểu enum 36](#_Toc126692007)

[Vòng lặp 37](#_Toc126692008)

[For 37](#_Toc126692009)

[Cú pháp 37](#_Toc126692010)

[Rút gọn 38](#_Toc126692011)

[Lược bỏ biểu thức khởi tạo 38](#_Toc126692012)

[Lược bỏ biểu thức điều kiện 38](#_Toc126692013)

[Lược bỏ biểu thức thay đổi 39](#_Toc126692014)

[Biến đêm 39](#_Toc126692015)

[While 40](#_Toc126692016)

[Cú pháp 40](#_Toc126692017)

[Rút gọn 41](#_Toc126692018)

[Do-while 41](#_Toc126692019)

[Hàm 41](#_Toc126692020)

[Nạp chồng hàm trong C++ (overload) 42](#_Toc126692021)

[Hàm nội tuyến (inline fuction) 43](#_Toc126692022)

[Đặt vấn đề 43](#_Toc126692023)

[Khai báo 44](#_Toc126692024)

[Tác dụng: 44](#_Toc126692025)

[Nguyên lý 44](#_Toc126692026)

[Hạn chế 45](#_Toc126692027)

[Ưu điểm: 45](#_Toc126692028)

[Nhược điểm: 45](#_Toc126692029)

[Áp dụng 46](#_Toc126692030)

[Con trỏ hàm 46](#_Toc126692031)

[Khai báo 46](#_Toc126692032)

[Gán địa chỉ 46](#_Toc126692033)

[Sử dụng 46](#_Toc126692034)

[Hàm con trỏ 47](#_Toc126692035)

[Một số hàm khác 48](#_Toc126692036)

[Strstr() 48](#_Toc126692037)

[strcpy() 49](#_Toc126692038)

[Pow() 49](#_Toc126692039)

[Sqrt() 50](#_Toc126692040)

[Putchar() 51](#_Toc126692041)

[Rand() 51](#_Toc126692042)

[Round() 53](#_Toc126692043)

[Làm tròn n chữ số sử dụng round() 54](#_Toc126692044)

[Tham khảo 55](#_Toc126692045)

# Tổng quan C++

(Thông tin từ website <http://laptrinhcanban.com/cpp/lap-trinh-cpp-co-ban/gioi-thieu-ngon-ngu-cpp/ngon-ngu-lap-trinh-cpp-la-gi/#Mot-so-dich-vu-duoc-phat-trien-bang-C>)

## C++ là ngôn ngữ lập trình mở rộng của C

C++ là một ngôn ngữ lập trình hướng đối tượng được mở rộng từ ngôn ngữ C. Trong ngôn ngữ C không có khái niệm lập trình hướng đối tượng, và để có thể lập trình hướng đối tượng thì các nhà phát triển đã thêm chức năng này cùng với nhiều cải tiến khác vào ngôn ngữ C, để tạo ra phiên bản mở rộng của C chính là ngôn ngữ C++.

Có thể hiểu đơn giản C++ chính là ngôn ngữ C với nhiều cải tiến khác nhau, bao gồm cả lập trình hướng đối tượng. Mặc dù được cải tiến từ C, nhưng nó vẫn có khả năng sử dụng các phương thức lập trình thủ tục như ngôn ngữ C truyền thống. Và một điều tuyệt vời nữa là, do C++ và C có tính tương hỗ với nhau, nên chúng ta cũng có thể sử dụng đồng thời cả C++ và C khi viết một chương trình máy tính.

## C++ là ngôn ngữ lập trình hướng đối tượng

Như đã nói ở trên thì ngôn ngữ lập trình C++ là một loại ngôn ngữ lập trình hướng đối tượng. Hướng đối tượng ở đây là ý tưởng kết hợp các đối tượng (sự vật) để chạy một chương trình, ví dụ để chạy một máy điều hòa không khí bạn cần kết hợp với một cái điều khiển từ xa chẳng hạn.

Mặc dù không phải là ngôn ngữ lập trình hướng đối tượng đầu tiên được phát triển trên thế giới, nhưng C++ cũng thuộc hàng cụ tổ cụ cố trong danh sách các loại ngôn ngữ lập trình hướng đối tượng. Thật vậy, C++ được phát triển vào năm 1983 bởi nhà khoa học Bjarne Stroustrup của Nokia Bell Labs như một phiên bản mở rộng của ngôn ngữ C, và nếu so với các loại ngôn ngữ lập trình hướng đối tượng khác như Java, Python hay Swift thì C++ xứng đáng được gọi là lão đại trong nghề lập trình rồi.

Tuy nhiên, cổ không có nghĩa là cũ. Do những ưu thế riêng biệt của mình mà cho đến tận ngày nay thì C++ vẫn giữ được độ hot của nó, và được các lập trình trên toàn thế giới yêu thích. Tại thời điểm Kiyoshi viết bài này (21-09-24) thì ngôn ngữ C++ cùng với ngôn ngữ tiền thân của nó là C đang cùng đứng thứ 6 trên bảng xếp hạng các ngôn ngữ lập trình được yêu thích trên toàn thế giới.

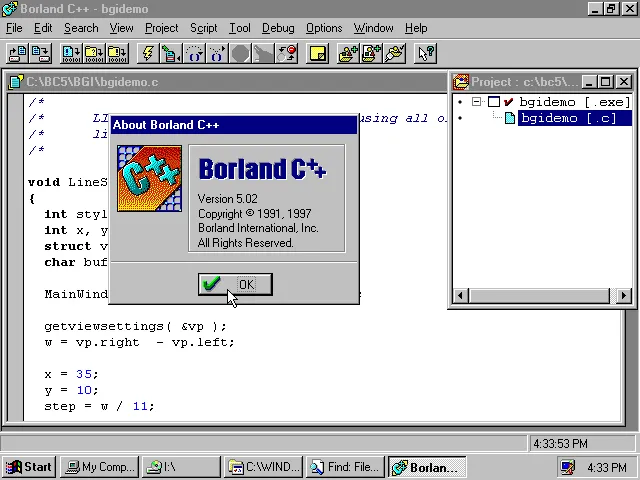
Do C++ được mở rộng từ C, nên tương tự như C thì ngôn ngữ C++ được thiết kế giống với cách con người nói và suy nghĩ, nhưng thứ con người nói thì máy tính lại không hiểu được. Do đó sau khi viết chương trình C++, để có thể chạy được chương trình này, chúng ta cần một hành động gọi là compile hay còn gọi là biên dịch, nhằm biên dịch mã nguồn của C++ thành dạng mã mà máy tính có thể hiểu và thực thi nó trong chương trình.

Và máy biên dịch sử dụng ở trên được gọi là trình biên dịch, và chúng ta sẽ cùng tìm hiểu trong một bài viết tiếp theo.

## Có nhiều loại môi trường phát triển C++ khác nhau

Ngôn ngữ C++ có rất nhiều loại môi trường phát triển khác nhau, và nổi tiếng nhất có thể kể đến là BorlandC++ và VisualC++.

BorlandC++ là một package của C++ được công ty Borland phát triển, và được kèm theo miễn phí môi trường phát triển tích hợp C++Builder.



VisualC++ (Microsoft Visual C ++ Plus) là loại C++ khác được phát triển trong môi trường phát triển tích hợp của Microsoft.

VisualC++ tuân thủ theo tiêu chuẩn Standard C ++ và được bán dưới dạng gói Visual Studio, là một môi trường phát triển được tích hợp với các ngôn ngữ phát triển khác nhau như Visual Basic và Visual C#.

## Các phiên bản của ngôn ngữ C++

C ++ có một lịch sử lâu đời, và nó cũng có rất nhiều phiên bản được cải tiến khác nhau xuyên suốt lịch sử của nó.

Trước khi C++ chính thức được đặt tên vào năm 1983, thì phiên bản đệm của C++ là C with Classes đã bắt đầu từ những năm 1979. Năm 1983, phiên bản này mới được đổi thành C++ cùng với nhiều cải tiến khác nhau đã thêm vào tại thời điểm đó. Và vào năm 1985, ấn bản đầu tiên của Ngôn ngữ lập trình C++ cũng đã được xuất bản thành sách.

Thời gian sau đó, C++ tiếp tục cải thiện và vào năm 2011, C ++ 11 đã được phát hành dưới dạng bản cập nhật lớn. Thời điểm hiện tại, phiên bản mới nhất là của C++ là C++17, được chính thức phát hành vào năm 2017.

Các phiên bản phát triển của C++ cụ thể như bảng dưới đây:



Figure 1. Các phiên bản của C++

Ngoài việc các tính năng khác nhau giữa các phiên bản C++, thì trong phiên bản mới nhất C++17 thì còn có rất nhiều các tính năng mới được thêm vào. Do đó, nếu bạn muốn học C++ thì Kiyoshi khuyên bạn lựa chọn các phiên bản mới như C++17 hoặc C++14 để học.

## C++ kế thừa ưu điểm về tốc độ xử lý của C

Giống như C thì C++ có khả năng xử lý chương trình cực nhanh. C++ hoạt động tương đối nhanh so với các ngôn ngữ script thông thường, và thích hợp để phát triển các ứng dụng thực hiện các xử lý phức tạp và cần tải cao cũng như phần mềm kinh doanh quy mô lớn.

Điều này là do C++ được kế thừa khả năng điều khiển trực tiếp bộ nhớ máy tính và tiết kiệm bộ nhớ của ngôn ngữ C.

Do đó, đối với các hệ thống xử lý lượng dữ liệu lớn, chẳng hạn như xử lý phân tích dữ liệu và machine learning, yêu cầu xử lý tốc độ cao thì ngôn ngữ C++ thường được lựa chọn để phát triển.

Giống như ngôn ngữ tiền thân là C thì C++ thường được sử dụng trong các hệ thống quy mô lớn trong suốt lịch sử phát triển của nó. Có thể nói C++ với sự tích hợp của nhiều công nghệ xử lý thông tin trong nó đã khiến C++ có độ tin cậy cao, và trở thành lựa chọn hàng đầu trong nhiều dự án.

Ứng dụng của C++ rất đa dạng, từ việc phát triển phần mềm, hệ thống cho đến các công nghệ tiên tiến như IoT và phân tích dữ liệu, và có thể nói đây là một ngôn ngữ lập trình rất đáng để chúng ta học hỏi.

## Hiểu về C++ sẽ hiểu về máy tính

Mặc dù không thể điều khiển trực tiếp phần cứng trong máy tính như ngôn ngữ assembly, nhưng C++ là ngôn ngữ lập trình bậc cao có khả năng thực hiện các xử lý điều khiển đồng mức như ngôn ngữ assembly.

Và vì C ++ có thể sử dụng tất cả các chức năng của ngôn ngữ C, nên nó đòi hỏi kiến ​​thức sâu hơn về máy tính so với các ngôn ngữ khác, chẳng hạn như con trỏ và các chức năng quản lý và tiết kiệm bộ nhớ khác.

Do vậy, bạn chắc chắn sẽ làm quen và hiểu sâu hơn về máy tính sau khi học C++, vì ngôn ngữ này có khả năng điều khiển cũng như cách sử dụng hiệu quả các tài nguyên có hạn của máy tính.

## Khả năng áp dụng vào các ngôn ngữ lập trình khác

Ngôn ngữ C++ là sự kết hợp hoàn hảo giữa tính linh hoạt của lập trình hướng đối tượng, và kế thừa tính cấp cao có thể được lệnh trực tiếp đến máy tính của ngôn ngữ C. Vì vậy, nếu bạn thành thạo ngôn ngữ C++ thì khi học các ngôn ngữ khác sẽ dễ dàng hơn.

Ngoài ra, với lợi thế quen thuộc hơn về máy tính, cũng như việc quen thuộc với cách viết các quy trình phức tạp khi bạn học C++, bạn sẽ ngạc nhiên về sự dễ hiểu của các ngôn ngữ khác và sẽ khiến việc học các ngôn ngữ khác sau khi học C++ trở nên dễ dàng hơn.

## Ứng dụng và trò chơi trên điện thoại thông minh

Để phát triển ứng dụng điện thoại thông minh, chúng ta có ngôn ngữ Java cho Android, Objective-C và Swift cùng một số ngôn ngữ khác cho iOS.

Tuy nhiên ngay cả trong C++, nếu bạn sử dụng môi trường phát triển tích hợp của C++ là C++ Builder, thì bạn có thể phát triển các ứng dụng gốc hỗ trợ cả Android lẫn iOS.

Bạn cũng có thể phát triển C++ trên Windows bằng cách sử dụng Visual Studio, môi trường phát triển tích hợp của Microsoft. C++ cũng được sử dụng trong phát triển trò chơi, hầu hết các trò chơi được làm bằng C hoặc C ++, và trong C++ cũng có các thư viện riêng chuyên cho việc phát triển trò chơi.

## Phát triển hệ thống kinh doanh

C++ thường được sử dụng trong các hệ thống kinh doanh quy mô lớn được vận hành bởi các công ty lớn, trong các hệ thống tài chính và hệ thống các tổ chức công do chính phủ điều hành.

Điều này là do C++ có khả năng kiểm soát các hệ điều hành phổ biến như Windows và Linux, cũng như tốc độ thực thi của các chương trình được viết bởi C++ nhanh hơn nhiều so với các ngôn ngữ khác.

Một lý do khác là do tiền thân của C++ là ngôn ngữ C vốn đã được sử dụng trong các hệ thống quy mô lớn, và C++ cũng được thừa hưởng độ tin cậy cao từ C, với việc được tích hợp nhiều công nghệ để xử lý thông tin.

## Ứng dụng WEB

C++ cũng được sử dụng trong việc phát triển một số ứng dụng web.

Có ý kiến ​​cho rằng C++ với tốc độ thực thi ứng dụng nhanh hơn và hoạt động ổn định so với các ngôn ngữ khác thì nên được sử dụng nhiều hơn trong phát triển ứng dụng web. Tuy nhiên thì gần đây các ngôn ngữ như PHP, Ruby và Python, vốn được tối ưu hóa để phát triển dịch vụ Web, mới đang là xu hướng chủ đạo.

Do đó, có thể nói rằng C++ hiếm khi được sử dụng để phát triển các ứng dụng web ngày nay.

## Một số dịch vụ được phát triển bằng C++

Có thể bạn không để ý, nhưng một số dịch vụ hay ứng dụng mà chúng ta hay sử dụng hằng ngày lại được phát triển bằng C++ đấy.

### Chrome

Trình duyệt web được sử dụng rộng rãi trên toàn thế giới này, và ngoài Cốc Cốc ra thì nó cũng đang được cài đặt trong hầu hết các máy tính kết nối mạng tại Việt Nam.



Phần lớn các ứng dụng, plugin của Chrome đều được viết bởi C++.

### Cocos2d-x

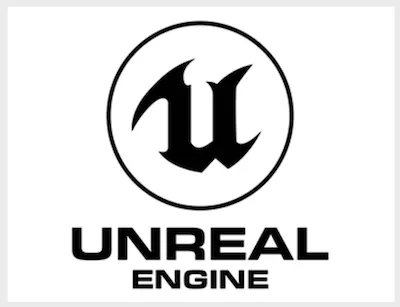
Cocos2d-x là một nền tảng đa năng được phát triển bởi ngôn ngữ C++, và được sử dụng để tạo ra các trò chơi chất lượng cao trên nhiều nền tảng khác nhau.



Ngoài C++, nó cũng có thể phát triển bằng các ngôn ngữ khác như JavaScript và Lua.

### Unreal-Engine

Unreal-Engine là một công cụ trò chơi được phát triển bởi C++, và được dùng để tạo ra các trò chơi chất lượng cao trên nhiều nền tảng khác nhau. Nó cũng được sử dụng trong việc phát triển các tựa game lớn.



# Hello world!

## Cấu trúc cơ bản của chương trình C++

#include <iostream>

using namespace std;

int main(){

  //Các câu lệnh xử lý

  return 0;

}

Thư viện iostream là thư viện chuẩn của C++

Namespace là không gian tên. Nếu không sử dụng **using namespace std** thì các câu lệnh viết như sau: std::cout

## Quy tắc viết chương trình C++

Khi viết chương trình trong C++, chúng ta cần tuân thủ một số các quy tắc nhằm xác định cách thức để viết và dịch trong ngôn ngữ lập trình C++ như sau:

Kết thúc câu lệnh bằng dấu chấm than ;

Sử dụng thụt lề (các khoảng trắng đầu câu lệnh) cũng như xuống dòng để làm cấu trúc chương trình dễ hiểu hơn. Tham khảo: Thụt lề trong C++

Sử dụng ký tự // để comment một dòng lệnh, và cặp dấu /\* và \*/ để comment nhiều dòng lệnh. Tham khảo: Comment trong C++

Bao gồm cả hàm main(), chương trình trong C++ được tạo tạo ra bởi các hàm.

Bắt đầu hàm bởi dấu { và kết thúc hàm bởi dấu }

## Hello world!

Chương trình đầu tiên, in ra dòng hello world!

#include <iostream>

using namespace std;

int main(){

  cout << "Hello World!\n";

  return 0;

}

Hello World!

## Comment

Comment một dòng: // <nội dung comment>

// dòng comment

Comment nhiều dòng: /\* <nội dung comment> \*/

/\*

dòng comment 1

dòng comment 2

…

\*/

# Nhập xuất

## 4 lệnh nhập xuất cơ bản trong C++

| **Lệnh** | **Ý nghĩa** |
| --- | --- |
| cin | Luồng đầu vào chuẩn |
| cout | Luồng đầu ra tiêu chuẩn |
| cerr | Luồng đầu ra tiêu chuẩn để ghi lỗi |
| clog | Luồng đầu ra tiêu chuẩn để ghi log |

## Toán tử nhập xuất

| **Toán tử** | **Ý nghĩa** | **Sử dụng trong lệnh** |
| --- | --- | --- |
| >> | toán tử nhập | cin |
| << | toán tử xuất | cout cerr clog |

## Cout

cout trong C++ hay còn gọi là luồng đầu ra tiêu chuẩn là một đối tượng trong class istream, được dùng để nhận dữ liệu từ trong chương trình để xuất ra bên ngoài chương trình đó.

VD1:

#include <iostream>

using namespace std;

int main() {

    /\*Xuất dữ liệu bằng lệnh cout\*/

    cout << "Xin chào";

    return 0;

}

Xin ch├áo

In ra với cout có thể bị lỗi font tiếng việt.

## Cerr

Dùng in ra lỗi

VD3

#include <iostream>

using namespace std;

int main() {

    int a = 233;

    char str[100] = "hello";

    // Xuất dòng lệnh thông báo lỗi

    cerr << "a :" << a << ", str: " << str << endl;

    cerr << "This is error" << endl;

    return 0;

}

a :233, str: hello

This is error

Hoạt động của cerr không khác Cout là mấy.

## Clog

Dùng in ra log

VD4

#include <iostream>

using namespace std;

int main() {

    int a = 233;

    char str[100] = "hello";

    // Xuất dòng lệnh thông báo lỗi

    clog << "a :" << a << ", str: " << str << endl;

    clog << "This is a log" << endl;

    return 0;

}

a :233, str: hello

This is a log

Hoạt động không khác Cout và Cerr.

## Cin

cin trong C++ hay còn gọi là luồng đầu vào tiêu chuẩn là một đối tượng trong class istream, được dùng để lưu giữ các dữ liệu đầu vào từ người dùng dưới dạng ký tự trong bộ nhớ máy tính.

VD2

#include <iostream>

using namespace std;

int main() {

    char name[100];

    //Nhập dữ liệu bằng lệnh cin

    cout << "Enter name:"; cin >> name;

    int a;

    cout << "Enter a:"; cin >> a;

    //Xuất dữ liệu bằng lệnh cout

    cout << "name :" << name << endl;

    cout << "a: " << a << endl;

    return 0;

}

Enter name:Hao

Enter a:12

name :Hao

a: 12

Cin nhận dữ liệu nhập từ bàn phím và ép về đúng với kiểu của biến.

Do cin là một đối tượng của class iostream nên cũng có các hàm thành viên, trong đó có cin.ignore và cin.getline.

### Cin.ignore

cin.ignore trong C++ là một hàm thành viên trong class istream có tác dụng bỏ qua các ký tự đầu vào cho đến khi đã trích xuất đủ số ký tự chỉ định, hoặc là cho tới khi tìm thấy ký tự phân tách được chỉ định.

Hàm cin.ignore sẽ dừng trích xuất ký tự nếu đã đến cuối dữ liệu (EOF). Và nếu hàm dừng trước khi trích xuất đủ số ký tự hoặc tìm thấy ký tự chỉ định, flag chứa thông tin lỗi sẽ được trả về.

Chúng ta sử dụng hàm cin.ignore trong C++ với cú pháp như sau:

cin.ignore( n, delim);

Trong đó n là số ký tự cần trích xuất, và delim là ký tự phân tách cần tìm.

#include <iostream>

using namespace std;

int main () {

    char first, last;

    cout << "Nhập tên họ cách nhau bởi dấu cách:\n";

    first = cin.get();         //Dùng hàm get để lấy ký tự đầu tiên

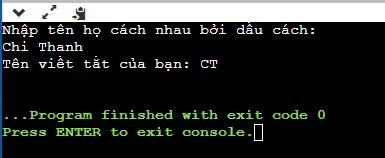
    cin.ignore(256,' ');  //Bỏ qua cho tới khi tìm thấy dấu cách

    last = cin.get();      //Dùng hàm get để lấy ký tự tiếp theo

    cout << "Tên viết tắt của bạn: " << first << last << '\n';

    return 0;

}



Trong ví dụ trên, khi nhập chuỗi Chi Thanh, cin.get() sẽ lấy ra ký tự đầu tiên ‘C’. Sau đó, cin.ignore() sẽ bỏ qua các kí tự tiếp theo cho đến khi gặp dấu cách. Cuối cùng cin.get() lấy ra kí tự tiếp theo sau dấu cách ‘T’.

### Cin.getline

cin.getline trong C++ là một hàm thành viên trong class istream có tác dụng trích xuất các ký tự từ luồng đầu vào tiêu chuẩn (cin) và lưu lại dưới dạng chuỗi ký tự, cho tới khi tìm thấy ký tự phân tách được chỉ định, hoặc cho đến khi đã trích xuất đủ số ký tự chỉ định.

Hàm in.getline sẽ dừng trích xuất ký tự nếu đã đến cuối dữ liệu (EOF). Và nếu hàm dừng trước khi trích xuất đủ số ký tự hoặc tìm thấy ký tự phân tách chỉ định, flag chứa thông tin lỗi sẽ được trả về.

cin.getline( str, n [, delim]);

Trong đó str là biến dùng để lưu kết quả trích xuất, n là số ký tự cần trích xuất, và delim là ký tự phân tách.

Lưu ý delim có thể được lược bỏ khi dùng hàm cin.getline trong C++. Khi đó, chương trình sẽ dùng giá trị mặc định là ký tự xuống dòng \n để quyết định khi nào dừng hàm.

#include <iostream>

using namespace std;

int main () {

    char name[256], title[256];

    cout << "Nhập tên bạn: ";

    cin.getline (name,256);

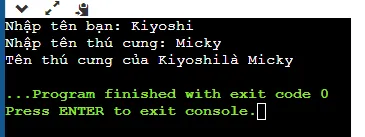
    cout << "Nhập tên thú cưng: ";

    cin.getline (title,256);

    cout << "Tên thú cưng của "<<name << " là "<< title;

    return 0;

}



# Toán tử

## Bảng toán tử so sánh

| **Toán tử** | **Biểu thức điều kiện** | **Ý nghĩa** |
| --- | --- | --- |
| == | x == y | x và y bằng nhau |
| != | x != y | x và y không bằng nhau |
| > | x > y | x lớn hơn y |
| < | x < y | x nhỏ hơn y |
| >= | x >= y | x bằng hoặc lớn hơn y |
| <= | x <= y | x bằng hoặc nhỏ hơn y |

## Bảng toán tử logic

| **Toán tử** | **Tên** | **Biểu thức** | **Ý nghĩa** |
| --- | --- | --- | --- |
| && | Toán tử và (AND) | X && Y | true nếu cả X và Y đều đúng |
| || | Toán tử hoặc (OR) | X || Y | true nếu ít nhất một trong hai vế X hoặc Y đúng |
| ! | Toán tử phủ định (NOT) | ! X | true nếu X sai và false nếu X đúng |

### Thứ tự ưu tiên toán tử logic trong C++

NOT > AND > OR

Ví dụ cụ thể thì ở dưới đây, toán tử AND && sẽ được đánh giá trước so với toán tử OR ||

4>3 || 2<3 && 1>2

//true

Để thay đổi thứ tự ưu tiên các toán tử này trong biểu thức, chúng ta sử dụng tới cặp dấu ngoặc () như sau:

(4>3 || 2<3) && 1>2

//false

## Bảng toán tử số học

| **Toán tử** | **Ví dụ** | **Ý nghĩa** |
| --- | --- | --- |
| + | A + B | Thêm B vào A |
| - | A -B | Trừ B cho A |
| \* | A \* B | Nhân A với B |
| / | A / B | Chia A cho B |
| % | A% B | Dư của phép chia A cho B |

## Toán tử tăng và giảm

| **Toán tử** | **Ví dụ** | **Ý nghĩa** |
| --- | --- | --- |
| ++ | A++ hoặc ++A | Tăng A thêm 1 đơn vị |
| – | A– hoặc –A | Giảm A xuống 1 đơn vị |

### Sự Khác Biệt Giữa ++ i Và i ++

Chúng ta có thể đặt các toán tử tăng và giảm tại phía trước hoặc phía sau đối tượng. Ví dụ các cách viết sau đều đưa ra kết quả giống nhau:

int x = 5;

cout << x++ << endl; //6

int x = 5;

cout << ++x << endl; //6

Tuy nhiên, trong trường hợp sau đây thì kết quả của chúng lại hoàn toàn khác nhau.

Đặt toán tử ở phía sau:

int i = 5;

int k;

k = i++;

cout << i << endl;  // 6

cout << k << endl;  // 5

Đặt toán tử ở phía trước:

int i = 5;

int k;

k = ++i;

cout << i << endl;  // 6

cout << k << endl;  // 6

Để hiểu rõ Sự Khác Biệt Giữa ++ i Và i ++ và tại sao kết quả phép tính ở trên lại khác nhau thì chúng ta cần hiểu rõ ở đây:

* Nếu đặt toán tử phía sau đối tượng (i++) thì trước hết giá trị của i được gán vào k trước, rồi sau đó mới tăng 1 đơn vị cho i. Nói cách khác k = i++ chính bằng phép toán k = i; => i = i +1;.
* Ngược lại thì nếu đặt toán tử phía trước đối tượng (++i) thì trước hết giá trị của i được tăng 1 đơn vị, rồi sau đó mới được gán vào k. Nói cách khác k = ++i chính bằng phép toán i = i +1; => k = i;.

## Toán tử 3 ngôi

condition ? true\_value : false\_value;

Trong đó condition là biểu thức điều kiện, true\_value là giá trị trả về khi điều kiện đúng và false\_value là giá trị trả về khi điều kiện sai.

VD

#include <string>

int n = 8;

string x;

x = (n == 10) ? "OK": "NG" ;

## Bảng toán tử gán

| **Toán tử** | **Ví dụ** | **Ý nghĩa tương đương** | **Ý nghĩa** |
| --- | --- | --- | --- |
| += | A += B | A = A + B | Cộng |
| -= | A -= B | A = A - B | Trừ |
| \*= | A \*= B | A = A \* B | Nhân |
| /= | A /= B | A = A / B | Chia |
| %= | A %= B | A = A % B | Lấy dư |
| &= | A &= B | A = A & B | Bit AND |
| |= | A |= B | A = A | B | Bit OR |
| ^= | A ^= B | A = A ^ B | Bit NOT |
| <<= | A <<= B | A = A << B | Dịch bit qua trái |
| >>= | A >>= B | A = A >> B | Dịch bit qua phải |

## Phép gán đồng thời

VD

int a, b, c;

a = b = c = 8;

Về bản chất của phép gán đồng thời này là chúng ta sẽ gán giá trị vào một biến, rồi dùng giá trị của biến này để gán vào các biến còn lại như sau:

int a, b, c;

c = 8;

b = c;

a = b;

# Kí tự

## Mã ký tự

Trong các bài trước chúng ta đã biết, C++ là một ngôn ngữ bậc cao và nội dung viết trong đó rất giống với cách viết và suy nghĩ của con người. Tuy nhiên thì trong máy tính chỉ có 2 trạng thái tồn tại đó là có dòng điện chạy qua và không có dòng điện chạy qua, tương ứng với 2 số 0 và 1 nên chúng ta cần phải biên dịch chương trình C++ cho máy tính hiểu được.

Một cách tương tự, đối với máy tính thì các chữ cái, dấu hoặc chữ số như a, 8, ! mà chúng ta sử dụng trong đời sống hàng ngày đều là ký tự không thể hiểu được. Để máy tính có thể hiểu chúng, con người đã xây dựng những bảng mã ký tự mà trong đó, mỗi ký tự của ngôn ngữ con người sẽ được gán cho một số thứ tự trong bảng mã, và máy tính thay vì đọc ký tự thì sẽ đọc thứ tự của ký tự đó trong bảng mã. Ví dụ như chữ a thì là thứ 1, chữ o thì là thứ 3 chẳng hạn. Và chúng ta gọi các số thứ tự trong bảng mã này theo cách thân mật là mã ký tự trong C++.

## Lấy mã ASSCII

Để lấy mã ASSCII của một ký tự bất kỳ, chúng ta có thể ép kiểu ký tự về kiểu int như sau:

cout << int('a') << endl; // 97

cout << int('A') << endl; // 65

## Chuyển từ mã ascii sang ký tự

Ngược lại thì chúng ta cũng có thể chuyển từ mã ascii sang ký tự trong C++ bằng cách ép kiểu số về kiểu ký tự thông qua hàm char() với cú pháp sau đây:

cout << char(97) << endl; // a

cout << char(65) << endl; // A

## Biểu diễn ký tự

Ký tự được biểu diễn nằm trong dấu nháy đơn ‘’.

Các chữ tiếng việt có dấu (như ă, ơ, …) và các ký tự đặc biệt (@, #, …) không phải là ký tự mà là chuỗi.

## Chuỗi ký tự

Chuỗi ký tự được biểu diễn dưới dấu nháy kép “”.

Biểu diễn chuỗi ký tự bằng tổ hợp ký tự. VD: "abc" <==> 'a' + 'b' + 'c' + '\0'

## Ký tự đặc biệt

|  |  |  |
| --- | --- | --- |
| 1 | \a | Âm cảnh báo |
| 2 | \b | backspace |
| 3 | \n | ký tự xuống dòng (LF) |
| 4 | \r | ký tự xuống dòng (CR) |
| 5 | \f | Tách trang |
| 5 | \t | ký tự TAB |
| 5 | \t | ký tự TAB |
| 6 | \v | TAB dọc |
| 7 | \¥ | Ký tự tiền yên |
| 8 | \? | Dấu chấm hỏi |
| 9 | \\ | Dấu gạch chéo ngược |
| 10 | \' | dấu nháy đơn (‘) |
| 11 | \" | dấu nháy kép (“) |
| 12 | \0 | Null |
| 13 | \000 | ASCII ký tự hệ cơ số 8 |
| 14 | \xhh | ASCII ký tự hệ cơ số 16 |

# Số

## Số nguyên

Khi biểu diễn số nguyên trong C++, nếu số nguyên ở dạng số thập phân thì chúng ta sẽ ghi trực tiếp số thập phân đó trong chương trình C++. Tuy nhiên nếu số nguyên ở các dạng khác thì chúng ta cần phải thêm ký tự chỉ định vào đầu số đó, ví dụ như ở dạng bát phân thì ghi thêm số 0 vào đầu, hoặc thập lục phân thì ghi thêm 0x vào đầu chẳng hạn.

88   // thập phân

045  // bát phân

0x3D // thập lục phân

## Số thực

VD

3.14

1.07

20.88

## Hằng số

Sử dụng const để khai báo hằng số. Hằng số có giá trị không đổi trong chương trình. Nếu trong chương trình, có đoạn code làm thay đổi hằng số sẽ sinh ra lỗi vì kiểu const không cho phép thay đổi giá trị của biến.

## Kiểm tra số chẵn lẻ

### Cách 1: chia cho 2 lấy dư.

int check\_odd\_even(int n){

    //flag = 1 => số lẻ

    //flag = 0 => số chẵn

    int flag = 1;

    if( n % 2 == 0 ) flag= 0;

    return flag;

}

### Cách 2: and với 1

Kết quả phép toán bit AND giữa số lẻ với số 1 luôn bằng 1, nên sử dụng kết quả này chúng ta có thể tạo hàm kiểm tra số chẵn lẻ trong C++ như sau:

int check\_odd\_even\_bit(int n){

    //flag = 1 => số lẻ

    //flag = 0 => số chẵn

    int flag = 0;

    if( n & 1 == 1 ) flag= 1;

    return flag;

}

## Kiểm tra số nguyên

So sánh 2 giá trị làm tròn lên và làm tròn xuống.

Số nguyên trong C++ có khả năng biểu diễn bởi giá trị chính xác của nó, trong khi số thực thì có thể làm tròn lên hoặc làm tròn xuống thành các giá trị khác nhau.

Bằng cách sử dụng tính chất này, chúng ta có thể so sánh hai giá trị làm tròn lên và làm tròn xuống của số đã cho, và nếu hai giá trị này giống nhau thì số đã cho chính là một số nguyên.

int check\_odd\_even\_bit(int n){

    //flag = 1 => số lẻ

    //flag = 0 => số chẵn

    int flag = 0;

    if( n & 1 == 1 ) flag= 1;

    return flag;

}

## Kiểm tra số nguyên tố

Số nguyên tố là số tự nhiên lớn hơn 1 không phải là tích của hai số tự nhiên nhỏ hơn. Nói cách khác, số nguyên tố là những số chỉ có đúng hai ước số là 1 và chính nó.

Phương pháp: kiểm tra trong phạm vi lớn hơn 1 và nhỏ hơn số đã cho có tồn tại ước số nào không

#include <iostream>

using namespace std;

int isprime(int n){

    //flag = 0 => không phải số nguyên tố

    //flag = 1 => số nguyên tố

    int flag = 1;

    if (n <2) return flag = 0; /\*Số nhỏ hơn 2 không phải số nguyên tố => trả về 0\*/

    /\*Sử dụng vòng lặp while để kiểm tra có tồn tại ước số nào khác không\*/

    int i = 2;

    while(i <n){

        if( n%i==0 ) {

            flag = 0;

            break; /\*Chỉ cần tìm thấy 1 ước số là đủ và thoát vòng lặp\*/

        }

        i++;

    }

    return flag;

}

## Kiểm tra số âm, số dương

Phương pháp: so sánh với 0

int check\_negative\_positive\_number(int n){

    //flag = -1 => số âm

    //flag =  0 =>  số 0

    //flag =  1 => số dương

    int flag = 0;

    if (n >0 ) flag = 1;

    else if (n<0) flag = -1;

    return flag;

}

## Kiểm tra số chính phương

Số chính phương hay còn gọi là số hình vuông là số tự nhiên có căn bậc hai là một số tự nhiên, hay nói cách khác, số chính phương bằng bình phương của một số tự nhiên.

Cách 1: kiểm tra từ 0 đến n xem có tại số có bình phương bằng n hay không.

#include <cmath>

int find\_square\_number(int n){

    //flag = 1 => số chính phương

    //flag = 0 => không phải số chính phương

    int flag = 0;

    //Tìm số bất kỳ nhỏ hơn hoặc bằng n mà bình phương bằng n

    int i = 0;

    while(i <= n){

        if( pow( i, 2) == n ) {

            flag = 1;

            break; /\*Chỉ cần tìm thấy 1 số là đủ và thoát vòng lặp\*/

        }

        i++;

    }

    return flag;

}

Cách 2: Kiểm tra xem căn bậc 2 của n có phải là số tự nhiên hay không.

#include <cmath>

/\*Hàm kiểm tra một số có phải là số nguyên hay không\*/

int check\_real\_integer\_number(float n){

    //flag = 1 => số nguyên

    //flag = 0 => số thực

    int flag = 1;

    if (ceil(n) != floor(n)) flag = 0;

    return flag;

}

/\*Hàm kiểm tra một số có phải là số chính phương hay không\*/

int find\_square\_number(int n){

    //flag = 1 => số chính phương

    //flag = 0 => không phải số chính phương

    int flag = 0;

    //Kiểm tra căn bậc 2 của số đó có phải là số nguyên hay không

    if (check\_real\_integer\_number (sqrtf(n))) flag = 1;

    return flag;

}

## Kiểm tra số hoàn hảo

Một số được coi là số hoàn hảo trong C++ nếu như tổng các ước nguyên dương chính thức của nó (ngoại trừ nó) bằng chính nó.

Phương pháp: tìm ra và tính tổng tất cả các ước số, rồi so sánh xem tổng này có bằng với số đã cho hay không.

int find\_perfect\_num(int n){

    // flag = 1 => số hoàn hảo

    // flag = 0 => không phải số hoàn hảo

    int flag = 0, total=0;

    //Tìm tổng ước số

    for (int i=1;i<n; i++){

         if (n % i == 0) total+=i;//Nếu là ước số thì cộng vào tổng

    }

    //So sánh tổng ước số với số đã cho để tìm ra số hoàn hảo

    if (total == n & n!=0) flag = 1;

    return flag;

}

## Tìm số đảo ngược

VD số đảo ngược của 1234 là 4321

Phương pháp: viết lại từng hàng trong số theo thứ tự ngược lại

/\*Hàm tìm số đảo ngược trong C++\*/

int reverse\_num(int n){

  int reverse = 0;

  while (n > 0) {

    reverse = reverse \* 10 + n % 10;

    n /= 10;

  }

  return reverse;

}

## Kiểm tra số đối xứng

Số nguyên n gọi là số đối xứng nếu đọc từ trái qua phải, hay từ phải qua trái đều được số giống nhau. Ví dụ: 11,121,101 là các số đối xứng.

Phương pháp: đảo ngược số đã cho và kiểm tra số thu về có bằng với số đó hay không.

/\*Hàm tìm số đảo ngược trong C++\*/

int reverse\_num(int n){

  int reverse = 0;

  while (n > 0) {

    reverse = reverse \* 10 + n % 10;

    n /= 10;

  }

  return reverse;

}

/\*Ứng dụng Hàm tìm số đảo ngược để tạo Hàm kiểm tra số đối xứng trong C++\*/

int symmetrical\_num(int n){

    // flag = 1 => số đối xứng

    // flag = 0 => không phải số đối xứng

    int flag =0;

    if (reverse\_num(n) == n) flag = 1;

    return flag;

}

## Số pi

### Sử dụng hằng số M\_PI trong cmath

VD

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    cout << M\_PI; //3.14159

}

### Sử dụng hằng số std::numbers::pi trong numbers

Từ phiên bản C++20 trở đi, chúng ta có thêm một phương pháp nữa để ký hiệu và hiển thị số pi trong C++, đó chính là sử dụng tới hằng số std::numbers::pi trong numbers.

VD

#include <iostream>

#include <numbers>

using namespace std;

int main() {

    cout << numbers::pi //3.14159

}

### Tự tạo số pi

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    double pi = atan(1)\*4;

    cout << pi; //3.14159

}

# Biến

## Khai báo

type name;

type name1, name2,name3, ...;

VD

//Biến thuộc kiểu số nguyên

int num1;

//Biến thuộc kiểu số thực

float num2;

//Biến thuộc kiểu ký tự

char txt;

//Biến thuộc kiểu chuỗi ký tự

std::string str;

int num1, num2, num3;

## Khởi tạo

name = value;

VD

int num;

num = 10;

int num1 = 10;

int num2 =11,num3= 12,num4 = 13;

# Lệnh điều kiện

## If

Cú pháp:

### If đơn

if (condition){

câu lệnh xử lý nếu condition là True (đúng)

}

Nếu chỉ có 1 câu lệnh trong khối if thì chúng ta cũng có thể lược bỏ cặp dấu {} và sử dụng if rút gọn trong C++ như sau:

if(condition) câu-lệnh;

### if else

if(condition){

câu lệnh xử lý nếu condition là True (đúng);

}else{

câu lệnh xử lý nếu condition là False (sai);

}

Nếu chỉ có 1 câu lệnh trong khối if else thì chúng ta cũng có thể lược bỏ cặp dấu {} và sử dụng if rút gọn trong C++ như sau:

if(condition) câu-lệnh-nếu-condition-là-True;

else câu-lệnh-nếu-condition-là-False;

### if elif else

if(condition-1){

câu lệnh xử lý nếu condition-1 là True (đúng);

}else if(condition 2){

câu lệnh xử lý nếu condition-2 là True (đúng);

}else if(condition 3){

câu lệnh xử lý nếu condition-3 là True (đúng);

…

}else{

câu lệnh xử lý nếu tất cả các condition ở trên đều False (sai);

}

## Switch case

### Cú pháp

switch (expression){

case value1：

//Xử lý nếu expression === value1

…

break;

case value2:

//Xử lý nếu expression === value2

…

break;

case value3:

//Xử lý nếu expression === value3

…

break;

default:

//Xử lý nếu giá trị của expression không khớp với một trong các giá trị chỉ định

…

}

### switch case trong C++ với kiểu enum

VD

#include <iostream>

using namespace std;

enum week {

  Mon,

  Tue,

  Wed,

  Thu,

  Fri,

  Sat,

  Sun

};

int main() {

  enum week day;

  day = Fri;

  switch (day) {

    case Mon:

      cout << "Thứ 2";

      break;

    case Tue:

      cout << "Thứ 3";

      break;

    case Wed:

      cout << "Thứ 4";

      break;

    case Thu:

      cout << "Thứ 5";

      break;

    case Fri:

      cout << "Thứ 6";

      break;

    case Sat:

      cout << "Thứ 7";

      break;

    case Sun:

      cout << "Chủ Nhật";

      break;

  }

  return 0;

}

//> Thứ 6

# Vòng lặp

## For

### Cú pháp

for (biểu-thức-khởi-tạo ; biểu-thức-điều-kiện ; biểu-thức-thay-đổi ) {

Câu lệnh 1 trong khối for ;

Câu lệnh 2 trong khối for ;

...

}

VD

#include <iostream>

using namespace std;

int main(){

    for (int i = 1; i < 3; i++) {

        cout << "hello "<< i <<endl;

    }

    cout << "bye";

    return 0;

}

### Rút gọn

Nếu chỉ có một câu lệnh duy nhất trong khối lệnh của for, chúng ta có thể lược bỏ cặp dấu {} và rút gọn vòng lặp for trong C++ như sau:

for (biểu-thức-khởi-tạo ; biểu-thức-điều-kiện ; biểu-thức-thay-đổi ) Câu lệnh;

VD

for (int i = 1; i < 3; i++) cout << "hello " << i <<endl;

### Lược bỏ biểu thức khởi tạo

Nếu chúng ta đã khởi tạo biến đếm ở ngoài vòng lặp for thì biểu thức khởi tạo có thể được rút gọn. Ví dụ như sau:

int i = 0;

for (; i < 2; i++){

  cout << "i=" << i <<endl;

}

### Lược bỏ biểu thức điều kiện

VD

int sum = 0;

for (int i = 1; ; i++){

  sum += i;

  if (sum > 5){

    break;

  }

}

cout << "sum = " << sum;

//sum = 6

### Lược bỏ biểu thức thay đổi

VD

int i = 1;

for (; i < 100;){

  cout << i<<endl;

  i \*= 3;

}

### Biến đêm

Phạm vi sử dụng: Trong vòng lặp for, nếu ra ngoài vòng lặp sẽ bị lỗi

VD

for (int i = 1; i < 3; i++) {

    cout << "hello "<< i <<endl;

}

cout << i;

// Lỗi biên dịch trả về

Nhiều biến đếm

Cú pháp:

for ( BTKT1 ; BTKT2 ; BTĐK ; BTTĐ1 ; BTTĐ2 ) {

Câu lệnh 1 trong khối for ;

Câu lệnh 2 trong khối for ;

…

}

VD

#include <iostream>

using namespace std;

int main(){

    for (int i = 1,j= 100; i < 5; i++, j--){

      cout << "i = "<< j << ",j = " <<j << endl;

    }

    return 0;

}

//> i = 1, j = 100

//> i = 2, j = 99

//> i = 3, j = 98

//> i = 4, j = 97

## While

### Cú pháp

while(biểu-thức-điều-kiện) {

Câu lệnh 1 trong khối while ;

Câu lệnh 2 trong khối while ;

...

}

VD

#include <iostream>

using namespace std;

int main(){

    int num = 1;

    while (num < 20){

        cout << num << endl;

        num = num \* 3;

    }

    return 0;

}

//1

//3

//9

### Rút gọn

Nếu chỉ có một câu lệnh duy nhất trong khối lệnh của while, chúng ta có thể lược bỏ cặp dấu {} và dùng while với cú pháp rút gọn như sau:

while(biểu-thức-điều-kiện) câu-lệnh;

## Do-while

Cú pháp:

do{

Câu lệnh 1 trong khối do…while ;

Câu lệnh 2 trong khối do…while;

…

}while( biểu-thức-điều-kiện );

Nếu chỉ có 1 câu lệnh duy nhất trong khối do…while thì chúng ta có thể lược bỏ đi cặp dấu ngoặc nhọn {} và sử dụng cú pháp sau:

do{

Câu lệnh ;

}while( biểu-thức-điều-kiện );

# Hàm

Hàm trong C++ là một tập hợp các xử lý nhằm thực hiện một chức năng cụ thể nào đó trong chương trình. Hàm cho phép bạn kết hợp các xử lý khác nhau thành một và đặt tên cho nó. Sau khi tạo và đặt tên cho một hàm, chúng ta chỉ cần gọi tên hàm ra mỗi khi cần sử dụng đến nó trong chương trình.

VD

#include <iostream>

using namespace std;

//Khai báo hàm

int find\_sum(int a,int b){

    int sum = a + b;

    return sum;

}

int main(){

    int sum = find\_sum(1,2); //Gọi hàm

    cout << sum;

    return 0;

}

//> 3

## Nạp chồng hàm trong C++ (overload)

Nạp chồng hàm cho phép chúng ta định nghĩa nhiều hàm với cùng một tên nhưng có khác nhau về các tham số đầu vào hay đầu ra.

Ví dụ cụ thể, 3 hàm sau đây tuy có cùng tên nhưng có signature khác nhau, dẫn đến việc nạp chồng hàm trong C++ có thể xảy ra, và chúng ta có thể sử dụng đồng thời chúng với cùng tên trong chương trình.

int sum(int a, int b) { ... }            // sum(int, int)

int sum(int a, int b, int c)  { ... }    // sum(int, int, int)

double sum(double a, double b)  { ... }  // sum(double, double)

int main()

{

    sum(2, 3);      // Hàm sum đầu tiên

    sum(2, 3, 4);   // Hàm sum thứ 2

    sum(2.0, 3.0);  // Hàm sum thứ 3

}

Ngược lại, nếu các hàm cùng tên mà lại có cùng signature thì việc nạp chồng hàm trong C++ không thể xảy ra, như các hàm sau đây:

int sum(int a, int b) { ...}

int sum(int x, int y) { ... }      // Dù tên đối số khác nhau nhưng có cùng signature

double sum(int a, int b) { ... }   // Dù kiểu trả về khác nhau nhưng có cùng signature

## Hàm nội tuyến (inline fuction)

Tham khảo:

Hkteam:<https://howkteam.vn/course/khoa-hoc-lap-trinh-c-can-ban/ham-noi-tuyen-trong-c-inline-functions-3894>

### Đặt vấn đề

Khi một hàm được gọi, CPU sẽ lưu địa chỉ bộ nhớ của dòng lệnh hiện tại mà nó đang thực thi (để biết nơi sẽ quay lại sau lời gọi hàm), sao chép các đối số của hàm trên ngăn xếp (stack) và cuối cùng chuyển hướng điều khiển sang hàm đã chỉ định. CPU sau đó thực thi mã bên trong hàm, lưu trữ giá trị trả về của hàm trong một vùng nhớ/thanh ghi và trả lại quyền điều khiển cho vị trí lời gọi hàm. Điều này sẽ tạo ra một lượng chi phí hoạt động nhất định so với việc thực thi mã trực tiếp (không sử dụng hàm).

Đối với các hàm lớn hoặc các tác vụ phức tạp, tổng chi phí của lệnh gọi hàm thường không đáng kể so với lượng thời gian mà hàm mất để chạy. Tuy nhiên, đối với các hàm nhỏ, thường được sử dụng, thời gian cần thiết để thực hiện lệnh gọi hàm thường nhiều hơn rất nhiều so với thời gian cần thiết để thực thi mã của hàm.

Trong C, chúng ta thường sử dụng hàm Macro, một kỹ thuật tối ưu hóa được sử dụng bởi trình biên dịch để giảm thời gian thực hiện. C++ cung cấp một khái niệm mới tốt hơn, đó là hàm nội tuyến (inline functions).

Inline functions (hàm nội tuyến) là một loại hàm trong ngôn ngữ lập trình C++. Từ khoá inline được sử dụng để đề nghị (không phải là bắt buộc) compiler (trình biên dịch) thực hiện inline expansion (khai triển nội tuyến) với hàm đó hay nói cách khác là chèn code của hàm đó tại địa chỉ mà nó được gọi.

### Khai báo

Thêm từ khoá “inline” và trước khai báo một hàm thông thường

VD

#include <iostream>

inline double square(double x)

{

    return x \* x;

}

int main()

{

    std::cout << square(3.0) << std::endl;   // 9

}

### Tác dụng:

Giúp chương trình xử lý nhanh hơn thông thường.

### Nguyên lý

Khi gọi một hàm,

thay vì: CPU phải thực hiện các bước như lưu địa chỉ bộ nhớ của dòng lệnh hiện tại mà nó đang thực thi (để biết nơi sẽ quay lại sau lời gọi hàm), gọi và thực thi hàm từ vùng nhớ của hàm, trở lại chương trình từ địa chỉ đã lưu.

thực hiện: các đoạn mã của hàm ngay trên bộ nhớ chương trình mà không cần phải mất chi phí cho việc gọi hàm.

Inline giúp trình biên dịch viết các đoạn mã của hàm ngay trong chương trình thực thi thay vì viết các đoạn mã của hàm tại một vùng nhớ khác rồi thực hiện các bước gọi hàm từ vùng nhớ đó. Việc này giảm bớt được chi phí gọi hàm nhưng sẽ khiến bộ nhớ chương trình tăng lên.

(Thông tin không chắc chắn) Inline tương tự macro khi sử dụng define trên ngôn ngữ C.

### Hạn chế

Trình biên dịch có thể không thực hiện nội tuyến trong các trường hợp như:

* Hàm chứa vòng lặp (for, while, do-while).
* Hàm chứa các biến tĩnh.
* Hàm đệ quy.
* Hàm chứa câu lệnh switch hoặc goto.

### Ưu điểm:

* Tiết kiệm chi phí gọi hàm.
* Tiết kiệm chi phí của các biến trên ngăn xếp khi hàm được gọi.
* Tiết kiệm chi phí cuộc gọi trả về từ một hàm.
* Có thể đặt định nghĩa hàm nội tuyến (inline functions) trong file tiêu đề (\*.h) (nghĩa là nó có thể được include trong nhiều đơn vị biên dịch, hàm thông thường sẽ gây ra lỗi).

### Nhược điểm:

* Tăng kích thước file thực thi do sự trùng lặp của cùng một mã.
* Khi được sử dụng trong file tiêu đề (\*.h), nó làm cho file tiêu đề của bạn lớn hơn.
* Hàm nội tuyến có thể không hữu ích cho nhiều hệ thống nhúng. Vì trong các hệ thống nhúng, kích thước mã quan trọng hơn tốc độ.

### Áp dụng

* Sử dụng với những hàm nhỏ nhưng phải thực hiện gọi hàm nhiều lần trong vòng lặp.
* Điều kiện: VXL có bộ nhớ chương trình lớn. Do sẽ làm tốn nhiều bộ nhớ chương trình.

## Con trỏ hàm

### Khai báo

type (\*fp) ( para\_type\_1, para\_type\_2, para\_type\_3,...);

VD

//Khai báo con trỏ hàm kiểu void với các đối số kiểu int

void (\*po1)(int, int);

//Khai báo con trỏ hàm kiểu int với các đối số kiểu int

int (\*po2)(int)

//Khai báo con trỏ hàm kiểu void của hàm không có đối số

void (\*po3)()

### Gán địa chỉ

fp = funct;

VD:

int add(int a, int b){

    return a + b;

}

int main(){

    int (\*fp)(int, int);  /\* Khai báo con trỏ hàm \*/

    fp = add;  /\* Gán địa chỉ hàm cho con trỏ \*/

}

### Sử dụng

(\*fp) ( para\_1, para\_2, para\_3,...);

VD

#include <iostream>

using namespace std;

//Khai báo hàm

int add(int a, int b)

{

    return a + b;

}

int main()

{

    int m = 2, n = 5, sum;

    int (\*fp)(int, int);  /\* Khai báo con trỏ hàm \*/

    fp = add;             /\* Gán địa chỉ hàm cho con trỏ \*/

    sum = (\*fp)(m, n);    /\* Gọi hàm bằng con trỏ hàm \*/

    cout << "sum = "<< sum;

    return 0;

}

## Hàm con trỏ

VD

#include <iostream>

using namespace std;

void swap(int \*x, int \*y) {

    int tmp;

    tmp = \*x;

    \*x = \*y;

    \*y = tmp;

}

int main() {

    int num1 = 123;

    int num2 = 456;

    cout << "Truoc khi hoan doi：num1 = "<< num1 <<", num2 = "<< num2 <<endl;

    swap(&num1, &num2);

    cout << "Sau khi hoan doi：num1 = "<< num1 <<", num2 = "<< num2;

    return 0;

}

Truoc khi hoan doi：num1 = 123, num2 = 456

Sau khi hoan doi：num1 = 456, num2 = 123

# Một số hàm khác

## Strstr()

Cú pháp

strstr(str, keyword);

Trong đó:

* str là chuỗi ban đầu
* keyword là chuỗi ký tự cần tìm trong chuỗi str.

Hàm strstr() sẽ trả về địa chỉ của vị trí đầu tiên tìm thấy của chuỗi ký tự keyword trong chuỗi str tính từ đầu chuỗi, và trả về vị trí này dưới dạng con trỏ chuỗi trong C++.

Cần khai báo thư viện cstring

VD

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str[30] = "Good School";

    char \* p;

    p = strstr(str, "oo");

    cout <<  p;

}

ood School

## strcpy()

Cú pháp:

strcpy(target, source);

Trong đó:

* targe là chuỗi đích
* source là chuỗi nguồn.

Hàm strcpy() sẽ copy toàn bộ nội dung của chuỗi source và phục chế nó vào chuỗi target.

Cần khai báo thư viện cstring

VD:

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str1[30];

    char str2[30] = "I love C++/C++";

    //Copy toàn bộ chuỗi str2 vào chuỗi str1

    strcpy(str1, str2);

    cout <<  str1;

}

//I love C++/C++

## Pow()

Cú pháp:

doule pow(doule x, doule y);

Trong đó x là cơ số, y là luỹ thừa, và doule là kiểu dữ liệu mặc định của cơ số và luỹ thừa sử dụng trong hàm pow.

Kết quả trả về của hàm pow sẽ là giá trị luỹ thừa mũ y của x với giá trị là xy.

VD:

#include<iostream>

#include<cmath>

using namespace std;

int main(){

    cout << "Luy thua 2 cua 3 la: "<< pow(2, 3) <<endl;

    cout << "Luy thua 3.1 cua 3 la: "<< pow(3.0, 3.1) <<endl;

    cout << "Luy thua 3.1 cua 3 la: "<< pow(2.012, 3.132) <<endl;

    return 0;

}

Luy thua 2 cua 3 la: 8

Luy thua 3.1 cua 3 la: 30.1353

Luy thua 3.1 cua 3 la: 8.93229

## Sqrt()

Tính căn bậc 2

double sqrt(double x);

Trong đó x là số thực thuộc kiểu double cần tìm căn bậc hai, và doule là kiểu dữ liệu mặc định của đối số cũng như giá trị trả về của hàm sqrt.

VD

#include <iostream>

#include <cmath>

using namespace std;

int main(){

    cout << sqrt(4) <<endl;

    cout << sqrt(100) <<endl;

    cout << sqrt(4.1) <<endl;

    return 0;

}

2

10

2.02485

## Putchar()

Dùng xuất 1 ký tự.

Putchar(char);

Trong đó char là ký tự (hoặc là biến chứa ký tự) mà chúng ta cần xuất ra màn hình.

VD:

#include <iostream>

using namespace std;

int main()

{

    putchar( 'a' );

    return 0;

}

A

## Rand()

Cú pháp:

srand((unsigned int)time(NULL));

rand();

Trong đó

* srand() đặt giá trị ban đầu cho rand. Sử dụng srand giúp có được trình tự ra các số ngẫu nhiên khác nhau từ các lần biên dịch khác nhau. Time() dùng lấy thời gian hiện tại.
* rand(): hàm lấy số ngẫu nhiên.

VD

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

//Tạo hàm GetRandom để chỉ định phạm vi sinh số ngẫu nhiên

int GetRandom(int min,int max){

    return min + (int)(rand()\*(max-min+1.0)/(1.0+RAND\_MAX));

}

int main(){

    //Sử dụng hàm srand để thay đổi số nguồn sử dụng trong hàm rand

    srand((unsigned int)time(NULL));

    for (int i = 0;i < 10;i++) {

        cout << GetRandom(1,6)<<endl;

    }

    return 0;

}

// kết quả chạy lần 1

6

1

5

4

3

5

5

2

4

5

// kết quả chạy lần 2

6

1

4

2

2

5

4

4

1

1

Xem chi tiết: <http://laptrinhcanban.com/cpp/lap-trinh-cpp-co-ban/ham-trong-cpp/ham-rand-trong-cpp/>

## Round()

Ứng dụng: Làm tròn số

round(x);

Trong đó x là số cần làm tròn. Giá trị của x có thể thuộc bất kỳ kiểu số nào trong C++, ví dụ như là double, float hay long double chẳng hạn.

Hàm round sẽ trả về kết quả làm tròn là số nguyên gần nhất với số chỉ định. Điều đó có nghĩa số chỉ định sẽ được tròn lên 1 đơn vị nếu phần thập phân >= 0.5 và lượt bỏ phần thập phân nếu < 0.5.

VD

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

    int a = 11;

    cout << round(a) << endl;       //11

    float b = 11.16;

    cout << round(b) << endl;       //11

    float c = 11.66666;

    cout << round(c) << endl;       //12

    double d = 11.66666;

    cout << round(d) << endl;       //12

    long double e = 11.66666;

    cout << round(e) << endl;       //12

}

### Làm tròn n chữ số sử dụng round()

Phương pháp: nhân số cần làm tròn với , làm tròn bằng round, sau đó chia lại cho .

VD

#include <iostream>

#include <math.h>

using namespace std;

/\*Tạo hàm làm tròn n chữ số trong C++\*/

double take\_decimal\_number(double num, int n){

    int base = pow(10,n);

    double result = round(num \* base) /base;

    return result;

}

int main()

{

    double num = 124.4677;

    cout << take\_decimal\_number(num, 2) <<endl;

    cout << take\_decimal\_number(num, 3) <<endl;

    return 0;

}

# Tham khảo

Lập trình căn bản:

<https://laptrinhcanban.com/cpp/lap-trinh-cpp-co-ban/>