### **Meet Bjarne Stroustrup**

### **Hello, It’s nice to see you here**

#### Xin chào lập trình viên trẻ,

#### Chúc mừng bạn đã tham gia khóa học này..

#### Bây giờ bạn có thể nghĩ tại sao tôi chúc mừng bạn?

#### Tôi là Bjarne Stroustrup! Người tạo ra C ++.

#### Và vì bạn đã đăng ký khóa học này, tôi sẽ đảm bảo rằng bạn nắm vững tất cả các khái niệm và trở thành một nhà phát triển tốt hơn!

#### Vì vậy, xin chúc mừng trước..

### **A small tip**

#### Trước khi chúng ta tiếp tục, tôi muốn cung cấp cho bạn một mẹo nhỏ!

#### Chỉ cần mã với tôi!

#### Thực hiện tất cả các ví dụ với tôi để đạt được nhiều nhất từ khóa học này ..

### **Meet C++**

#### Tôi đã phát triển C ++ tại Bell Labs vào năm 1980.

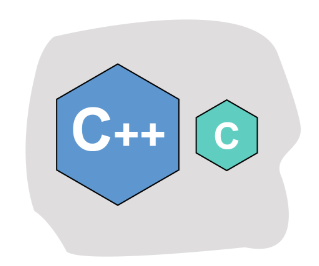
#### Nó rất giống với Lập trình C mà người bạn đời của tôi, Dennis Ritchie đã tạo ra vào đầu những năm 1970.

#### Tôi đã phát triển C ++ như một phần mở rộng cho ngôn ngữ C.

#### Tôi đã phát triển C ++ để cung cấp các cơ sở của Simula cho việc tổ chức chương trình cùng với hiệu quả và tính linh hoạt của C cho lập trình hệ thống.

#### C++ tương thích với C đến mức nó có thể biên dịch hơn 99% chương trình C mà không thay đổi một dòng mã nguồn.

#### Mặc dù, C ++ là một ngôn ngữ có cấu trúc tốt và an toàn hơn nhiều so với C vì nó dựa trên OOP.



#### Tôi sẽ gọi C++ là siêu tập hợp của C!

#### Cũng đúng nếu chúng ta gọi C ++ là "C với các lớp".

#### Ngoài ra, nếu bạn không biết một số điều khoản mà chúng ta đang thảo luận, đừng lo lắng, tôi ở đây vì bạn!

#### Đến cuối khóa học, bạn sẽ thành thạo với tất cả các khái niệm này!

### **See you in the next section**

#### Này bạn thân,

#### Đi chậm! Đó là khá nhiều nó cho phần này!

#### Bây giờ bạn đã biết điều gì đó về C ++.

#### Vì vậy, hãy hạnh phúc, và bước vào phần tiếp theo với sự tự tin hơn!

#### Cho đến lúc đó, Taddaa!

### **What is C++? Introduction to C++**

#### Vì vậy, chào mừng bạn trở lại phần này!

#### Trong phần này, chúng ta sẽ đi sâu vào C ++

#### Vì vậy, hãy bắt đầu

#### C++ là một ngôn ngữ lập trình hướng đối tượng có mục đích chung.

#### Theo mục đích chung, ý tôi là nó có nhiều trường hợp sử dụng khác nhau và được sử dụng trong các khu vực rộng lớn.

#### Bây giờ, ngôn ngữ lập trình hướng đối tượng là gì?

#### Nó chỉ là một khái niệm đơn giản hoặc một cách viết một chương trình mà chúng ta sử dụng các lớp và đối tượng.

#### Chúng ta sẽ đi sâu vào vấn đề này trong phần nâng cao.

#### Bây giờ, hãy bám sát những điều cơ bản

### **Know a little more**

#### Bạn muốn biết thêm một chút về người bạn mới của bạn C ++?

#### C ++ là một ngôn ngữ cấp trung, vì nó gói gọn cả các tính năng ngôn ngữ cấp cao và cấp thấp.

#### Ngoài ra, C ++ hỗ trợ cả mô hình lập trình hướng đối tượng và thủ tục; do đó C++ còn được gọi là ngôn ngữ lai.

#### Bây giờ, bạn đã có đủ ý tưởng về C + + là gì.

#### Vì vậy, hãy tiến lên phía trước.

### **Why C++?**

### **I developed C++ in 1980!**

#### Vâng, bạn đã nghe nó ngay trong 1980..

#### Đã 40 năm trôi qua và C++ vẫn còn mạnh mẽ ở đó!

#### Theo Chỉ số ngôn ngữ lập trình Tiobe, C++ được xếp hạng 4 tính đến tháng 2020 năm <>.

#### Điều này có thúc đẩy bạn học C ++ không?

### **Why learn C++?**

#### Chà, hãy để tôi cung cấp cho bạn thêm một số lý do để học C ++.

* C ++ là PHẢI cho sinh viên và các chuyên gia làm việc để trở thành một Kỹ sư phần mềm tuyệt vời.
* C ++ rất gần với phần cứng, vì vậy bạn có cơ hội làm việc ở mức thấp, cho phép bạn kiểm soát rất nhiều về mặt quản lý bộ nhớ, hiệu suất tốt hơn và cuối cùng là phát triển phần mềm mạnh mẽ.
* Lập trình C++ cung cấp cho bạn một sự hiểu biết rõ ràng về Lập trình hướng đối tượng.
* C++ là một trong những ngôn ngữ lập trình thường xanh và được hàng triệu nhà phát triển phần mềm yêu thích.
* C++ là ngôn ngữ lập trình được sử dụng rộng rãi nhất trong lập trình ứng dụng và hệ thống.

### **Applications of C++**

#### Một số hệ thống được sử dụng dễ thấy nhất hiện nay có các phần quan trọng được viết bằng C ++.

* Bloomberg (hình thành tài chính),
* Amazon (Thương mại web), Google (Tìm kiếm trên web)
* Facebook (truyền thông xã hội)

#### Nhiều ngôn ngữ lập trình phụ thuộc vào hiệu suất và độ tin cậy của C ++ trong việc triển khai chúng. Các ví dụ bao gồm:

* Máy ảo Java
* Trình thông dịch JavaScript (ví dụ: phiên bản V8 của Google)
* Trình duyệt (ví dụ: Internet Explorer, Firefox của Mozilla, Safari của Apple và Chrome của Google)
* Khung ứng dụng và Web (ví dụ: khung dịch vụ Web .NET của Microsoft).

### **More on C++**

#### Hơn C++ được sử dụng trong,

* C++ được sử dụng để tạo trò chơi và đồ họa.
* C++ được sử dụng trong các máy ATM mà bạn rút tiền mặt.
* C ++ cũng được sử dụng để tạo các ứng dụng máy tính để bàn mà chúng ta sử dụng hàng ngày.

#### Vì vậy, bạn đã quyết định biến C ++ thành bạn của bạn và tôi làm cố vấn của bạn cho đến khi kết thúc khóa học?

#### Đó là tinh thần..

#### Hẹn gặp lại các bạn trong phần tiếp theo.

### **How does C ++ work?**

### **Understanding the working of C++**

#### Vậy ngôn ngữ lập trình là gì?

#### Một khái niệm thông qua đó chúng ta viết lệnh cho các máy tính để chúng có thể hiểu và thực hiện một tác vụ cụ thể. Bên phải?

#### Bây giờ, bạn phải tự hỏi làm thế nào các lệnh mà chúng ta viết bằng tiếng Anh giống như ngôn ngữ được máy tính hiểu và làm thế nào nó hoạt động phù hợp?

#### Hãy để tôi cố gắng giải thích điều này trong năm bước đơn giản.



1. Đầu tiên, chúng tôi viết các lệnh bằng ngôn ngữ C ++ dựa trên tiếng Anh
2. Sau đó, các lệnh dựa trên tiếng Anh được gửi đến một "Trình biên dịch"
3. Công việc của trình biên dịch là chuyển đổi các lệnh dựa trên tiếng Anh thành các số chứa số một và số không (1s và 0s), ở dạng nhị phân
4. Những con số này sau đó được đưa vào máy tính
5. Máy tính đọc các lệnh này và thực hiện các tác vụ.

#### Do đó, đây là cách chương trình C ++ hoạt động.

### **Let’s check how much you know**

### **Test Time**

#### Bạn thân, bây giờ bạn đã sẵn sàng để tiếp tục và viết một số mã trong C ++.

#### Nhưng trước đó, hãy để tôi làm một bài kiểm tra nhỏ và sau đó trong chủ đề tiếp theo, dạy bạn viết một số mã.

### **Well done!**

#### Thế là đủ rồi nhóc!

#### Bạn đã tiến bộ tốt.

#### Hãy nghỉ ngơi một chút và tham gia cùng tôi trong chương tiếp theo, nơi tôi sẽ dạy bạn viết một số mã!

#### Cho đến lúc đó, hãy nhớ rằng, cần có thời gian để trở thành một lập trình viên giỏi.

### **Our first C++ code**

### **Welcome back**

#### Bạn đã đến được chương 2!

#### Tôi hy vọng bạn đang tận hưởng công ty của tôi?

#### Cho đến bây giờ chúng tôi đã nghiên cứu rất nhiều lý thuyết, và hãy để tôi nói với bạn đời, điều đó là cần thiết!

#### Bây giờ trong phần này cuối cùng sự chờ đợi của bạn đã kết thúc ...

#### Bạn sẽ viết mã C ++ đầu tiên của mình trong phần này

### **Get ready**

#### Vì vậy, hãy viết một số mã...

#### Sẵn sàng?

#### Cố lên...

#### Đừng gian lận, hãy chuẩn bị sẵn sàng cho máy của bạn, mở trình soạn thảo mã mà bạn muốn, chỉ cần viết mã với tôi.

### **Legendary Hello World Program**

#### Tôi không có tâm trạng phá vỡ truyền thống, vì vậy hãy bắt đầu với chương trình Hello World huyền thoại.

#include <iostream>

using namespace std;

int main() {

cout << "Hello World!";

return 0;

}

Hello World

### **Let’s break it down**

#### Vâng, bạn thân, tôi biết bạn đang hạnh phúc...

#### Tôi biết cảm giác đó, khi bạn viết chương trình đầu tiên của mình và bạn nhận được 'Hello World' được in lên màn hình.

#### Vì vậy, hãy thử và hiểu code đầu tiên của chúng ta.

#### Hãy chia nhỏ code để hiểu rõ hơn:

#### **Dòng 1**: #include <iostream> là một thư viện tệp tiêu đề cho phép chúng ta làm việc với các đối tượng đầu vào và đầu ra, chẳng hạn như cout (được sử dụng trong dòng 5). Các tệp tiêu đề thêm chức năng cho các chương trình C ++.

#### **Dòng 2**: sử dụng namespace std có nghĩa là chúng ta có thể sử dụng tên cho các đối tượng và biến từ thư viện chuẩn.

#### Đừng lo lắng nếu bạn không hiểu cách #include <iostream> và sử dụng không gian tên std hoạt động.

#### Chỉ cần nghĩ về nó như một cái gì đó (gần như) luôn xuất hiện trong chương trình của bạn.

#### **Dòng 3**: Một thứ khác luôn xuất hiện trong chương trình C++ là int main().

#### Đây được gọi là một hàm.

#### Bất kỳ mã nào bên trong dấu ngoặc nhọn {} của nó sẽ được thực thi.

#### hàm main()là điểm khởi đầu của bất kỳ chương trình C++ nào.

#### **Dòng 4**: cout (phát âm là "see-out") là một đối tượng được sử dụng cùng với toán tử chèn (<<) để xuất / in văn bản. Trong ví dụ của chúng tôi, nó sẽ xuất ra "Hello World".

#### Lưu ý: Mỗi câu lệnh C++ kết thúc bằng dấu chấm phẩy (;)

#### **Dòng 5**: trả về 0 kết thúc hàm chính.

#### Bất cứ khi nào kiểu trả về của một hàm không bị vô hiệu, chúng ta trả về một số giá trị .

#### Trong ví dụ này, return type là int và chúng ta trả về 0.

#### Một lần nữa, đừng lo lắng, chúng tôi sẽ nghiên cứu sâu tất cả những điều này sau này trong khóa học.

#### **Dòng 6**: Đừng quên thêm dấu ngoặc nhọn đóng } để thực sự kết thúc chức năng chính.

### **A task**

#### Vì vậy, đó là khá nhiều cho mã đầu tiên của chúng tôi.

#### Vì vậy, tôi cho rằng bạn đã thành thạo với nội dung mà chúng ta đã học được ngay bây giờ và bây giờ là lúc giao cho bạn một số nhiệm vụ và sau đó bắt kịp bạn trong phần tiếp theo.

#### Viết mã in tên của bạn trong màn hình đầu ra.

#### (Gợi ý: Bạn cũng có thể sử dụng trình biên dịch có sẵn trên ứng dụng của chúng tôi)

### **Comments- What are they?**

### **Comments**

#### Này, tôi hy vọng bạn đã hoàn thành nhiệm vụ mà tôi đã giao cho bạn trong phần trước.

#### Vì vậy, hãy tiến về phía trước.

#### Trong phần này, chúng ta sẽ tìm hiểu về các ý kiến.

#### Bình luận là gì?

#### Chúng có liên quan đến những bình luận mà chúng ta làm trên phương tiện truyền thông xã hội của mình không?

#### Này nhóc, trước khi bạn mở rộng trí tưởng tượng của mình, hãy tiếp tục.

### **In simple terms**

#### Chúng tôi bình luận về một số hình ảnh trên phương tiện truyền thông xã hội phải không?

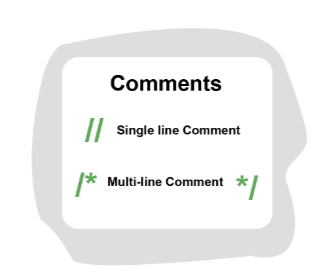
#### Tại sao chúng ta làm điều đó? Để mô tả một cái gì đó về hình ảnh đó phải không?

#### Tương tự như vậy trong lập trình, chúng tôi nhận xét để mô tả mã của chúng tôi hoặc để cung cấp thêm một số chi tiết về mã.

### **Comments in C++**

#### Nhận xét chương trình là các tuyên bố giải thích mà bạn có thể đưa vào mã C ++.

#### Những nhận xét này giúp bất cứ ai đọc mã nguồn. Tất cả các ngôn ngữ lập trình cho phép một số hình thức nhận xét.



#### C++ hỗ trợ nhận xét một dòng và nhiều dòng.

#### Tất cả các ký tự có sẵn bên trong bất kỳ nhận xét nào đều bị trình biên dịch C ++ bỏ qua.

#### Nhận xét một dòng bắt đầu bằng //

#### Ví dụ:

// This is a comment

#### Nhận xét nhiều dòng bắt đầu bằng /\* và kết thúc bằng \*/

#### Ví dụ:

/\* This is a multi line comment

We are learning C++ \*/

### **Example**

#### Khi đoạn mã trên được biên dịch, nó sẽ bỏ qua // in Hello World và tệp thực thi cuối cùng sẽ tạo ra kết quả sau:

Hello World

### **Our old custom**

#### Vì vậy, tôi hy vọng phần này thú vị vì bạn đã viết mã C ++ đầu tiên của mình.

#### Vì vậy, không vi phạm phong tục, chúng ta hãy có một bài kiểm tra nhỏ.

#### Bạn đã làm đủ trong phần này bạn đời...

#### Hãy nghỉ ngơi!

#### Tôi sẽ gặp bạn trong phần tiếp theo.

### **Understanding how to store data in C++**

### **You are great!**

#### Này, bạn đã là một người học giỏi.

#### Đã đến lúc tiếp tục và tìm hiểu một số khái niệm thú vị trong chương này.

#### Bám sát cho đến cuối cùng và thực thi mọi đoạn mã nhỏ mà chúng tôi sẽ thực hiện!

#### Vì vậy, hãy bắt đầu ...



#### Máy tính có không gian bộ nhớ lớn

#### Nhưng là lập trình viên, chúng ta cần sử dụng những không gian này một cách hiệu quả.

#### Không giống như phòng ngủ của chúng tôi, chúng tôi không muốn không gian này trông lộn xộn!



#### Do đó, chúng tôi sử dụng khái niệm lưu trữ đồ đạc trong các thùng chứa.

#### Nó tương tự như cách chúng ta giữ thực phẩm trong hộp đựng bên trong tủ lạnh.

#### Những hộp đựng này giúp bạn dễ dàng truy cập đồ đạc / thực phẩm bất cứ khi nào chúng ta cần.

### 

### **Let me introduce you to variables**

#### Bây giờ hãy giả sử bạn cần lưu trữ một cái gì đó trong bộ nhớ máy tính của bạn.

#### Đối với điều này, bạn sẽ cần một hộp đựng, giống như cái bạn có trong tủ lạnh.

#### Trong lập trình, container lưu trữ này được gọi là **biến**.

#### Vì vậy, hãy nhớ từ "biến" vì chúng ta sẽ sử dụng tên này rất nhiều từ bây giờ.

#### Việc xác định thực phẩm trong hộp đựng sẽ trở nên dễ dàng hơn nhiều nếu bạn đã dán nhãn lên hộp đựng trong tủ lạnh.

#### Tương tự, chúng ta phải cung cấp bộ chứa lưu trữ của mình, còn được gọi là biến với một **unique name**, để nó đến với dịch vụ của chúng ta bất cứ khi nào được gọi.

#### Cùng với đó, chúng ta cũng cần đề cập đến **type of value** mà nó sẽ chứa, ví dụ: từ, số, v.v.

### **Need of a variable**

#### Bây giờ bạn có thể tự hỏi, loại dữ liệu nào chúng ta cần lưu trữ và chính xác lý do tại sao chúng ta cần lưu trữ nó.

#### Thay vào đó, nó là bất kỳ loại ứng dụng nào, một ứng dụng web đơn giản, nơi chúng tôi sẽ cần lưu trữ thông tin đăng nhập hoặc có thể là một trò chơi nơi chúng tôi sẽ lưu trữ điểm số trò chơi, chúng tôi sử dụng các biến để lưu trữ những thứ này!

#### Do đó, hiểu rằng các biến có thể là một khái niệm đơn giản, nhưng là một khái niệm mạnh mẽ.

#### Để hiểu rõ hơn, hãy so sánh các bước lưu trữ thực phẩm trong hộp đựng để lưu trữ dữ liệu trong một biến.

1. Xác định một container: xác định loại biến
2. Cung cấp cho vùng chứa một nhãn: gán tên cho biến
3. Cho thức ăn vào hộp đựng : đặt một giá trị vào biến.

#### Đó là nó! Nó đơn giản như vậy.

#### Hãy xem các bước này sẽ trông như thế nào trong C ++;

* Xác định loại biến: Giả sử chúng ta muốn lưu trữ tên của bạn và tên của bạn sẽ là một từ, và một từ trong lập trình được gọi là chuỗi! Chúng ta sẽ đi sâu vào vấn đề này trong các phần sắp tới.
* Cung cấp cho container một label: firstName (tên chúng ta đang đặt cho container, MẸO: Hãy chắc chắn rằng các tên mà bạn đặt cho container phải có ý nghĩa! Đó là một thực hành tốt.)
* Đặt một giá trị trong biến : firstName = "John" ()chúng ta đã gán John làm giá trị để lưu trữ trong container.

#### Đây là cách cuối cùng nó sẽ trông như thế nào:

string firstName = “John”;

#### Bây giờ hãy in giá trị

cout << firstName;

John

#### Xin chúc mừng, bạn đã học được cách lưu trữ một giá trị trong một biến, vì vậy bây giờ chúng ta hãy đi sâu vào!

### **Types of Data in C++**

### **Data Types**

#### Vì vậy, bây giờ bạn đã học cách lưu trữ dữ liệu trong C ++.

#### Bây giờ hãy để tôi đưa bạn đi trước trong cuộc hành trình này với một trong những khái niệm thực sự quan trọng, **Kiểu dữ liệu**.

#### So, let’s stick to our refrigerator example.

#### Chúng ta cần các thùng chứa khác nhau để lưu trữ các mặt hàng khác nhau trong tủ lạnh của chúng ta.

#### Bên phải?

#### Cũng giống như cách bạn có các thùng chứa nước và sữa khác nhau, ngôn ngữ lập trình máy tính hoặc bất kỳ hệ thống kỹ thuật số nào nói chung đều có các "thùng chứa" khác nhau để chứa các loại thông tin khác nhau mà nó đi qua.

#### Các "loại vùng chứa" này tạo ra các kiểu dữ liệu khác nhau.

### **Kiểu dữ liệu**

#### Điều này cho phép tách các loại hoạt động có thể được thực hiện trên các phần thông tin khác nhau .

#### Bạn muốn có thể thêm giá của một cây bút chì với giá của một cây bút. Do đó, giá được xác định là một loại dữ liệu cụ thể.

#### Trong khi tên không được thêm vào số học.

#### Vì vậy, một hệ thống sẽ đặt nó trong một kiểu dữ liệu khác với giá.

### **Data Types in C++**

#### Sau đây là các kiểu dữ liệu cơ bản trong C ++ mà chúng ta sẽ tìm hiểu:

* Số nguyên (int) - Để lưu trữ số
* Dấu phẩy động (float) - Để lưu trữ số phân số / số thập phân
* Double (double) - Để lưu trữ các số phân số / thập phân với các điểm lớn
* Ký tự (char) - Để lưu trữ ký tự / chữ cái đơn
* Chuỗi (chuỗi) - Để lưu trữ các từ / chuỗi ký tự
* Boolean (bool) - Để lưu trữ các giá trị boolean, cụ thể là đúng và sai;

#### Bây giờ bạn đã biết các kiểu dữ liệu là gì và tại sao chúng ta cần chúng.

#### Ngoài ra bạn đã quen thuộc với các loại, do đó chúng ta hãy xem một ví dụ.

#### Giả sử chúng ta muốn tạo một ứng dụng để lưu trữ điểm số trò chơi.

#### Vì vậy, với kiến thức từ phần trước, hãy lặp lại các bước tương tự và khai báo một biến với kiểu dữ liệu thích hợp sẽ là Số nguyên, vì điểm số trò chơi sẽ là một giá trị số.

#### Vì vậy, mã của bạn sẽ trông giống như:

int gameScore = 0;

cout << gameScore;

#### Ở đây, int là kiểu dữ liệu, gameScore là tên biến và 0 là giá trị ban đầu được cung cấp cho biến.

#### Tương tự, chúng ta có thể sử dụng float, double, char, string để lưu trữ giá trị thích hợp theo yêu cầu.

#### Vì vậy, hãy tiếp tục và viết một số mã để thử các loại dữ liệu này!

### **What are booleans?**

#### Này, tôi giả sử rằng bạn đã thực hành các kiểu dữ liệu mà tôi đã đề cập trong màn hình trước!

#### Nào, hãy là một học sinh giỏi và thử chúng!

#### Bạn có câu hỏi?

#### Là câu hỏi của bạn - Boolean là gì?

#### Đừng lo lắng, tôi đã bảo vệ bạn.

### **What are booleans?**

#### Rất thường xuyên, trong lập trình, bạn sẽ cần một kiểu dữ liệu chỉ có thể có một trong hai giá trị, như:

* YES / NO
* ON / OFF
* TRUE / FALSE

#### Đối với điều này, C ++ có kiểu dữ liệu bool, có thể lấy các giá trị true (1) hoặc false (0).

#### Có một số tình huống mà chúng ta cần kiểm tra một cái gì đó, có thể lấy ra một giá trị có / không, đúng / sai.

#### Ở đây, booleans đi vào hành động

### **Example**

#### Trên đây là một ví dụ về cách chúng ta có thể sử dụng Booleans.

#### Vì vậy, đó là nó cho chủ đề này, tôi sẽ gặp bạn trong chủ đề tiếp theo!

#### Tadaaa...

### **Scopes of storing data**

### **What is a Scope?**

#### Này, tôi biết bạn đang nghĩ gì!

#### Tại sao chúng ta đang thảo luận về phạm vi trong một khóa học lập trình.

#### Chờ đã, đây không phải là phạm vi mà bạn thường nói đến trong trò chơi PUBG yêu thích của mình!

#### Chúng tôi đang thảo luận về phạm vi của các biến !!

### **Scope**

#### Phạm vi là một khu vực của chương trình và nói rộng ra có ba nơi, nơi các biến có thể được khai báo -

* Bên trong một hàm hoặc một khối được gọi là biến cục bộ,
* Trong định nghĩa của các tham số hàm, được gọi là tham số chính thức.
* Bên ngoài tất cả các hàm được gọi là biến toàn cục.

#### Chúng ta sẽ tìm hiểu hàm là gì và các tham số của nó trong các chương tiếp theo.

#### Ở đây hãy để chúng tôi giải thích các biến cục bộ và toàn cục là gì.

### **Local Scope**

#### Các biến được khai báo bên trong một hàm hoặc khối là các biến cục bộ.

#### Chúng chỉ có thể được sử dụng bởi các câu lệnh bên trong hàm hoặc khối mã đó.

#### Các biến cục bộ không được biết đến với các hàm bên ngoài của chúng.

#### Trên đây là một ví dụ về cách sử dụng biến cục bộ.

### **Global Scope**

#### Các biến toàn cục được định nghĩa bên ngoài tất cả các hàm, thường là trên đầu chương trình.

#### Các biến toàn cục sẽ giữ giá trị của chúng trong suốt vòng đời của chương trình của bạn.

#### Trên đây là một ví dụ về cách sử dụng biến cục bộ.

#### Một biến toàn cục có thể được truy cập bởi bất kỳ hàm nào.

#### Đó là, một biến toàn cục có sẵn để sử dụng trong toàn bộ chương trình của bạn sau khi khai báo.

### **Phewww…**

#### Đó là lý thuyết đủ.

#### Hãy xem một ví dụ.

### **Example**

#### Một chương trình có thể có cùng tên cho các biến cục bộ và toàn cục nhưng giá trị của biến cục bộ bên trong một hàm sẽ được ưu tiên.

#### Khi đoạn mã trên được biên dịch và thực thi, nó tạo ra kết quả sau:

10

### **Accepting user input**

### **Let’s bring the user in action**

#### Này nhà vô địch,

#### Cho đến bây giờ chúng tôi đã tạo ra các chương trình cứng nhắc!

#### Bây giờ là lúc để làm cho các chương trình của chúng tôi năng động ...

#### Đã đến lúc tạo một chương trình để người dùng có thể tương tác với nó.

### **User Input**

#### Thay vào đó, nó là bất kỳ loại ứng dụng nào, đầu vào của người dùng là thành phần chính!

#### Từ một ứng dụng dựa trên menu đơn giản đến một ứng dụng web chính thức, đầu vào của người dùng là thứ làm cho ứng dụng có ý nghĩa và tương tác hơn.

### **How?**

#### Bạn đã học được rằng cout được sử dụng để xuất (in) các giá trị.

#### Bây giờ chúng ta sẽ sử dụng cin để lấy đầu vào của người dùng.

#### Cin được phát âm là "see-in".

### **CIN**

#### Cin là một biến được xác định trước đọc dữ liệu từ bàn phím với toán tử trích xuất (>>).

#### Cú pháp tương tự như cú pháp của cout.

#### Hãy cố gắng hiểu với một ví dụ.

### **Example**

string name;

cout << “Enter your name:”

cin>>name;

cout<< “Name entered is:” << name;

#### Hãy cố gắng hiểu mã,

#### Đầu tiên chúng ta khai báo một tên biến của kiểu dữ liệu chuỗi sẽ chứa giá trị được nhập bởi người dùng.

#### Tiếp theo, chúng tôi in một tuyên bố tốt đẹp cho người dùng "Nhập tên của bạn:"

#### Sau đó, con trỏ sẽ đợi người dùng nhập một số giá trị vì chúng ta đã sử dụng cin.

#### Khi người dùng nhập giá trị, giá trị sẽ được lưu trữ trong biến name.

#### Cuối cùng, chúng tôi chỉ cần in giá trị do người dùng nhập.

### **Creating a dynamic calculator**

### **Calculator**

#### Chúng tôi đã làm điều này trước đây, phải không?

#### Bạn đã phát triển một mã để trừ hai số bằng cách sắp xếp lại chúng.

#### Và đó là niềm vui!

#### Nhưng, điều đó quá cứng nhắc...

#### Với các giá trị cố định, vì vậy hãy tạo một chương trình để thêm hai số với chức năng động bằng cách chấp nhận đầu vào từ người dùng.

### **The Code**

int x, y;

int sum;

cout << "Type a number: ";

cin >> x;

cout << "Type another number: ";

cin >> y;

sum = x + y;

cout << "Sum is: " << sum;

#### Trong chương trình trên, người dùng phải nhập hai số.

#### Sau đó, chúng tôi in tổng bằng cách tính (cộng) hai số.

#### Đó là khá nhiều nó!

#### Hẹn gặp lại các bạn trong chủ đề tiếp theo...

### **What is decision making?**

### **Why did you choose C++**

#### Tại sao bạn đăng ký khóa học này?

#### Vâng, bạn đọc nó đúng!

#### Tôi đang hỏi bạn điều này...

#### Có rất nhiều công nghệ trên thị trường.

#### Có rất nhiều khóa học trong ứng dụng của chúng tôi

#### Tại sao lại là C++?

#### Trước khi bạn nghĩ thêm...

#### Cuộc thảo luận của chúng ta sẽ hướng tới một khái niệm quan trọng trong lập trình.

#### Vì vậy, trước khi tham gia khóa học này, bạn có thể đã nghĩ đến một số điều kiện phải không?

#### Các điều kiện như tôi nên đăng ký C ++ hoặc C hoặc có thể là C ++ có giúp tôi nâng cao bộ kỹ năng của mình không?

#### Hoặc một số điều kiện khác.

#### Và tùy thuộc vào câu trả lời của họ, bạn có thể đã đưa ra một số quyết định.

#### Vì vậy, chúng ta đưa ra nhiều quyết định trong cuộc sống hàng ngày.

#### Bây giờ hãy để tôi cung cấp cho bạn một ví dụ phù hợp hơn.

### **Example**

#### Giả sử bạn muốn phát triển một số trò chơi

#### Và bạn muốn kiểm tra xem người dùng đã đăng nhập vào hồ sơ và sau đó chơi trò chơi hay chưa.

#### Vì vậy, chúng tôi muốn đưa ra một số quyết định tùy thuộc vào một số điều kiện.



#### Ở đây có khái niệm ra quyết định trong lập trình.

#### Ra quyết định là quyết định thứ tự thực hiện các tuyên bố dựa trên các điều kiện nhất định hoặc lặp lại một nhóm các tuyên bố cho đến khi các điều kiện cụ thể nhất định được đáp ứng.

#### Cấu trúc ra quyết định yêu cầu lập trình viên chỉ định một hoặc nhiều điều kiện được chương trình đánh giá hoặc kiểm tra, cùng với một tuyên bố hoặc câu lệnh sẽ được thực thi nếu điều kiện được xác định là đúng và tùy chọn, các câu lệnh khác sẽ được thực thi nếu điều kiện được xác định là sai.

#### C++ cung cấp tuyên bố ra quyết định sau:

* C++ cung cấp tuyên bố ra quyết định sau:
* Chuyển đổi trường hợp

#### Chúng tôi sẽ nghiên cứu những điều đó trong các phần tiếp theo!

#### Hẹn gặp bạn ở đó.

### **Decision making using if-else**

#### Về cơ bản nếu khác là một tuyên bố có điều kiện.

#### Bạn có thể sử dụng nếu một mình hoặc kết hợp với người khác.

#### Trong tiếng Anh nói chung, nếu khác, có thể được giải thích là cái này (như nếu) hoặc cái kia (như khác).

#### Nó được sử dụng trong điều kiện đơn giản mà chúng ta chỉ có 2 lựa chọn. if-else có thể được sử dụng hiệu quả bằng cách kết hợp chúng một cách thông minh.

#### Như trong sơ đồ trên, người ta có thể thấy rằng nếu điều kiện được cung cấp trong if là true thì tập hợp các mã dưới if block được thực thi.

#### Và nếu nó trở thành sai, thì tập hợp các câu lệnh trong khối Else sẽ được thực thi.

#### Và chương trình còn lại đi theo con đường.

### **Syntax**

#### Trên đây là cú pháp của khối if-else.

#### Nếu biểu thức boolean đánh giá là đúng, thì khối mã if sẽ được thực thi, nếu không khối mã khác sẽ được thực thi.

### **Example**

#### Hãy viết mã để kiểm tra tính đủ điều kiện bỏ phiếu.

#### GỢI Ý: Ở đây, chúng ta cần kiểm tra rằng nếu tuổi của ứng cử viên lớn hơn hoặc bằng 18 so với tuổi họ đủ điều kiện bỏ phiếu nếu không họ không thể

### **Code**

int age = 15;

if (age >= 18){

cout<< “You are eligible to vote”;

}

else{

cout<<”Nope, you cannot vote”;

}

Nope, you cannot vote

#### Trong ví dụ trước, chúng ta khai báo một độ tuổi biến đổi với giá trị 15.

#### Sau đó, chúng tôi kiểm tra điều kiện if, trong đó > = là dấu hiệu đại diện lớn hơn hoặc bằng, nếu điều kiện hóa ra là đúng, Bạn đủ điều kiện để bỏ phiếu được in.

#### Nếu điều kiện hóa ra là sai, thì điều khiển sẽ chuyển sang khối khác và Không, bạn không thể bỏ phiếu để chạy nước rút.

### **Decision making using switch case**

### **Switch Case**

#### Nó là cái gì?

#### Nói một cách đơn giản,

#### Một câu lệnh switch cho phép một biến được kiểm tra sự bình đẳng so với một danh sách các giá trị.

#### Mỗi giá trị được gọi là một trường hợp và biến đang được bật được kiểm tra cho từng trường hợp.

### **Syntax**

#### Đừng lo lắng, hãy để tôi giải thích cho bạn.

#### Các quy tắc sau đây áp dụng cho một câu lệnh chuyển đổi:

* Biểu thức được sử dụng trong câu lệnh switch là một số điều kiện hoặc biểu thức boolean.
* Bạn có thể có bất kỳ số lượng câu lệnh trường hợp nào trong một công tắc. Mỗi trường hợp được theo sau bởi giá trị được so sánh và dấu hai chấm.
* Biểu thức hằng số cho một trường hợp phải có cùng kiểu dữ liệu với biến trong công tắc và nó phải là hằng số hoặc nghĩa đen.
* Khi biến được bật bằng một trường hợp, các câu lệnh sau trường hợp đó sẽ thực thi cho đến khi đạt được câu lệnh break.
* Khi đạt đến câu lệnh ngắt, công tắc sẽ chấm dứt và luồng điều khiển chuyển sang dòng tiếp theo sau câu lệnh chuyển đổi.
* Không phải mọi trường hợp đều cần phải nghỉ ngơi. Nếu không có sự phá vỡ nào xuất hiện, luồng kiểm soát sẽ rơi vào các trường hợp tiếp theo cho đến khi đạt được sự phá vỡ.
* Câu lệnh chuyển đổi có thể có trường hợp mặc định tùy chọn, trường hợp này phải xuất hiện ở cuối công tắc. Trường hợp mặc định có thể được sử dụng để thực hiện một tác vụ khi không có trường hợp nào là đúng. Không cần nghỉ trong trường hợp mặc định.

### **Example**

#### Hãy phát triển một hệ thống tính điểm

You passed Your grade is D

#### Ở đây, biểu thức, nghĩa là biến cấp được kiểm tra dựa trên một vài điều kiện và theo đó các câu lệnh được thực thi.

### **A Quickie**

#### Này nhóc, tôi hy vọng bạn đang tận hưởng hành trình lập trình C ++ của mình ngay bây giờ và bạn háo hức muốn tiến lên phía trước.

#### Nhưng, tôi không thể phá vỡ truyền thống bạn đời!

#### Trước khi chúng ta tiếp tục, chúng ta hãy có một bản tóm tắt nhỏ

### **What is looping?**

### **Looping**

#### Chào mừng bạn đến với chủ đề mới!

#### Một lần nữa trong phần này, tôi sẽ cố gắng làm cho mọi thứ đơn giản nhất có thể.

#### Vì vậy, hãy bắt đầu,

#### Looping là gì?

#### Nói một cách đơn giản, lặp lại có nghĩa là lặp lại một số nhiệm vụ.



#### Một lần lặp lại là bình thường giống như trong cuộc sống thực.

#### Chẳng hạn

* Trong thư viện; Người ta có thể liên tục đọc thẻ danh mục để tìm vị trí của cuốn sách mà anh ta muốn đọc.

#### Xin hãy quên các danh mục máy tính, giả sử sử dụng thẻ danh mục tồn tại trên tủ vật lý.

#### Kỹ thuật lặp được sử dụng có thể khác nhưng nhiệm vụ đọc thẻ danh mục vẫn giống nhau, trong trường hợp tốt nhất, chỉ cần một tác vụ đọc.



* Tại nhà bếp; Người ta có thể liên tục cắt lát hành tây khi làm sốt pizza.

#### Nhiệm vụ cắt lát những củ hành tây đó được lặp đi lặp lại.



* Khắp nơi; Người ta có thể liên tục hát một bài hát trong nỗ lực ghi nhớ nó.

#### Nhiệm vụ để hát những bài hát tương tự lặp đi lặp lại là một sự lặp lại.

### **Looping in Programming**

#### Do đó, trong lập trình,

#### Có thể có một tình huống, khi bạn cần thực thi một khối mã nhiều lần.

#### Nói chung, các câu lệnh được thực hiện tuần tự:

#### Câu lệnh đầu tiên trong một hàm được thực thi đầu tiên, tiếp theo là câu lệnh thứ hai, v.v.

#### Ngôn ngữ lập trình cung cấp các cấu trúc điều khiển khác nhau cho phép các đường dẫn thực thi phức tạp hơn.

#### C++ cung cấp các câu lệnh điều khiển Loop sau đây mà chúng ta sẽ tìm hiểu trong các phần tiếp theo:

* Đối với vòng lặp
* Trong khi vòng lặp
* Làm trong khi lặp lại

### **For loop in C++**

### **For loop**

#### Hãy bắt đầu với cái đầu tiên: **Đối với vòng lặp**

#### Đối với vòng lặp

### **Syntax**

for ( init; condition; increment ) {

statement(s);

}

#### Luồng điều khiển trong vòng lặp for −

* Bước init được thực hiện đầu tiên và chỉ một lần.

#### Bước này cho phép bạn khai báo và khởi tạo bất kỳ biến điều khiển vòng lặp nào.

#### Bạn không bắt buộc phải đặt một câu lệnh ở đây, miễn là dấu chấm phẩy xuất hiện.

* Tiếp theo, điều kiện được đánh giá.

#### Nếu đó là sự thật, phần thân của vòng lặp được thực thi.

#### Nếu nó sai, phần thân của vòng lặp không thực thi và luồng điều khiển sẽ chuyển sang câu lệnh tiếp theo ngay sau vòng lặp for.

* Sau khi phần thân của vòng lặp for thực thi, luồng điều khiển sẽ nhảy trở lại câu lệnh tăng.

#### Câu lệnh này có thể để trống, miễn là dấu chấm phẩy xuất hiện sau điều kiện.

* Tình trạng này hiện được đánh giá lại.

#### Nếu đó là sự thật, vòng lặp thực thi và quá trình lặp lại chính nó (phần thân của vòng lặp, sau đó là bước tăng dần, và sau đó là điều kiện một lần nữa).

#### Sau khi điều kiện trở nên sai, vòng lặp for chấm dứt.

### **Example**

#### Khi đoạn mã trên được biên dịch và thực thi, nó in các số từ 10 đến 19

#### Đầu tiên, chúng ta khởi tạo một biến a với giá trị 10

#### Tiếp theo, chúng tôi đưa ra một điều kiện a< 20, điều này có nghĩa là giá trị của a không được lớn hơn hoặc bằng 20.

#### Cuối cùng, chúng ta tăng giá trị của biến a lên 1

#### Do đó, trong lần thực hiện đầu tiên,

#### Giá trị của a : 10

#### được in.

#### Sau đó, giá trị được tăng thêm một và kiểm tra xem nó có nhỏ hơn 20 hay không.

#### Quy trình tương tự tiếp tục cho đến khi điều kiện trở nên sai và điều khiển ra khỏi vòng lặp.

### **C++ while loop**

#### Vì vậy, bây giờ bạn đã biết vòng lặp là gì và làm thế nào cho các vòng lặp hoạt động!

#### Vì vậy, các chủ đề sắp tới trong phần này sẽ dễ dàng và nhanh chóng.

### **While loop**

#### Một câu lệnh vòng lặp thời gian liên tục thực hiện một câu lệnh mục tiêu miễn là một điều kiện nhất định là đúng.

#### Dưới đây là cú pháp của vòng lặp while,

while(condition) {

statement(s);

}

#### Ở đây, (các) câu lệnh có thể là một câu lệnh đơn lẻ hoặc một khối các câu lệnh.

#### Điều kiện có thể là bất kỳ biểu thức nào, và true là bất kỳ giá trị nào khác không.

#### Vòng lặp lặp lại trong khi điều kiện là đúng.

#### Khi điều kiện trở nên sai, điều khiển chương trình sẽ chuyển đến dòng ngay sau vòng lặp.

### **Example**

#### Trên đây là ví dụ tương tự mà chúng tôi đã thử với vòng lặp.

#### Lần này chúng tôi đã thực hiện nó bằng cách sử dụng một vòng lặp while!

#### Đầu ra vẫn giữ nguyên, đó chỉ là cách viết các vòng lặp...

### **Do while loop**

#### Này, đừng mệt...

#### Chỉ để tăng cường sự nhiệt tình của bạn, hãy để tôi nói với bạn đây là phần cuối cùng của chủ đề này và sẽ thực sự ngắn gọn và đơn giản!

#### Vì vậy, hãy bắt đầu ...

### **Do - While Loop**

#### Không giống như các vòng lặp for và while, kiểm tra điều kiện vòng lặp ở đầu vòng lặp, do... Trong khi vòng lặp kiểm tra tình trạng của nó ở dưới cùng của vòng lặp.

#### A làm... trong khi vòng lặp tương tự như vòng lặp while, ngoại trừ việc làm ... trong khi vòng lặp được đảm bảo thực hiện **ít nhất một lần**.

### **Syntax**

do {

statement(s);

}

while( condition );

#### Lưu ý rằng biểu thức có điều kiện xuất hiện ở cuối vòng lặp, vì vậy (các) câu lệnh trong vòng lặp thực thi một lần trước khi điều kiện được kiểm tra.

#### Nếu điều kiện là đúng, luồng điều khiển sẽ nhảy trở lại để thực hiện và (các) câu lệnh trong vòng lặp thực thi lại.

#### Quá trình này lặp lại cho đến khi điều kiện nhất định trở nên sai.

### **Example**

#### Trên đây là ví dụ tương tự mà chúng tôi đã làm trước đây, lần này sử dụng Do-while loop!

### **Knowledge check**

#### Vòng lặp thật thú vị...

#### Phải không?

#### Hãy kiểm tra xem bạn biết bao nhiêu.

#### Ok, bạn dường như đang trở nên tốt hơn,

#### Hãy tăng độ khó...

#### Bạn có thể viết mã để tìm giai thừa của một số bằng cách sử dụng vòng lặp while không?

#### GỢI Ý: Hàm giai thừa (ký hiệu: !) cho biết nhân tất cả các số nguyên từ số đã chọn của chúng ta xuống còn 1.

#### Ví dụ: 4! = 4 × 3 × 2 × 1 = 24

#### Thế là đủ rồi!

#### Hãy nghỉ ngơi, hãy hạnh phúc, bạn đã làm tốt!

#### Hẹn gặp lại các bạn trong phần tiếp theo.

### **What is an array?**

#### Hãy tưởng tượng một hộp bánh quy. Nếu bạn nhìn kỹ, bạn sẽ nhận thấy những điều dưới đây.

#### Hộp bánh quy là một thùng chứa tất cả các cookie.

#### Vì nó là một hộp cookie, chỉ có cookie được lưu trữ trong đó



#### Mỗi cookie ở vị trí duy nhất của nó.

#### Chúng tôi có thể cung cấp số cho cookie từ 1 đến 5

#### Mỗi chiếc bánh quy có thể được chọn theo vị trí số của nó, giống như nếu bạn muốn ăn bánh quy số 4 thì bạn sẽ nhặt bánh quy ở vị trí thứ 4.

#### Hãy tưởng tượng, bạn muốn tạo một loại hộp cookie như vậy trong lập trình sẽ giúp bạn lưu trữ các loại giá trị tương tự, tức là chỉ cookie.

#### Thứ hai, bạn có thể truy cập các cookie (giá trị) này bất cứ khi nào bạn muốn.

#### Làm thế nào để bạn nghĩ rằng điều này có thể đạt được trong C ++?

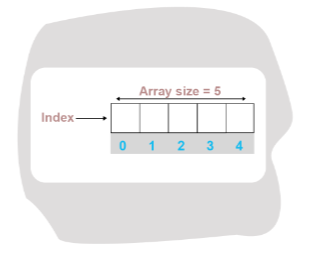
#### Câu trả lời là bằng cách sử dụng '**Arrays**'.

#### Hãy xem nó được thực hiện như thế nào.

### **What is an array?**

#### Mảng không là gì ngoài một container giúp bạn lưu trữ các loại giá trị tương tự, tức là các giá trị của **kiểu dữ liệu** tương tự.

#### Các giá trị này được gọi là Elements.



#### Bạn có thể truy cập từng Element theo vị trí duy nhất của nó trong mảng bất cứ khi nào bạn muốn.

#### Vị trí duy nhất này được gọi là **chỉ mục**.

#### Do đó, Một mảng trong C ++ là một tập hợp các mục được lưu trữ tại các vị trí bộ nhớ liền kề và các phần tử có thể được truy cập ngẫu nhiên bằng cách sử dụng các chỉ số của một mảng.

#### Chỉ số trong mảng luôn bắt đầu từ **0**.

#### Số lượng phần tử mà một mảng có thể lưu trữ được gọi là **Kích thước mảng**.

#### Vội! Bây giờ bạn đã biết một mảng là gì. Hãy xem cách khai báo một mảng trong C++,

#### Cú pháp:

type arrayName[arraySize];

#### **type** : Loại giá trị được lưu trữ trong một mảng, ví dụ: int, chuỗi, v.v.

#### **arrayName** : Tên của mảng

#### **arraySize** : số lượng giá trị tối đa có thể được lưu trữ trong một mảng.

#### Vì vậy, chúng ta hãy thử khai báo một mảng sẽ lưu trữ số cuộn của 5 sinh viên.

int roll\_no[5];

#### Đừng lo lắng, chúng ta sẽ tìm hiểu chi tiết điều này về cách nó hoạt động trong các phần tiếp theo.

### **Working with arrays**

### **Array Initialization**

#### Gán giá trị cho các phần tử của một mảng được gọi là khởi tạo mảng.

#### Hãy khởi tạo roll\_no mảng của chúng ta và lưu trữ số cuộn của 5 sinh viên.

roll\_no[0]=45;

roll\_no[1]=89;

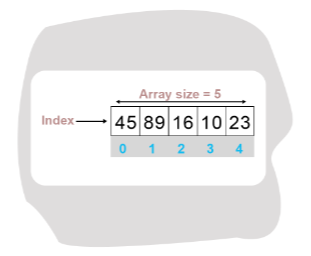
roll\_no[2]=16;

roll\_no[3]=10;

roll\_no[4]=23;

#### Các số bên trong dấu ngoặc vuông là chỉ số.

#### Chúng ta đã học được rằng mỗi phần tử của một mảng được truy cập bởi chỉ mục của nó và nó luôn bắt đầu từ 0.



#### Giá trị của các phần tử tại chỉ số 0 sẽ là 45, tại 1 sẽ là 89, v.v.

#### Lưu ý rằng chúng ta cũng có thể kết hợp khai báo mảng và khởi tạo như bên dưới'

int roll\_no[5]={45,89,16,10,23};

### **Accessing elements**

#### Như đã thảo luận trước đó, một phần tử trong một mảng được truy cập bởi chỉ mục của nó.

#### Chúng ta có thể sử dụng vòng lặp 'for' để truy cập các phần tử bằng cách lặp qua chỉ mục của nó.

int roll\_no[5]={45,89,16,10,23};

for (int i =0,i<5;i++){

cout<< “Element at index ”<<i<<”is”<<roll\_no[i]<<endl;

}

### **Output**

Element at index 0 is 45

Element at index 1 is 89

Element at index 2 is 16

Element at index 3 is 10

Element at index 4 is 23

#### Do đó, vòng lặp for sẽ lặp lại thông qua chỉ mục của mảng roll\_no và chúng ta truy cập các phần tử mảng bằng roll\_no[i].

#### Đó là về mảng!

### **What is a Function?**

### **Functions?**

#### Nếu bạn muốn một nhóm các dòng (viz, một mô-đun) trong C ++ được chạy đi chạy lại với các giá trị khác nhau, bạn phải sử dụng các hàm.

#### (Đối với các giá trị tương tự, bạn có thể sao chép dán, nhưng nó sẽ làm cho chương trình lộn xộn và khó đọc.)

#### Ví dụ: nếu bạn cần tạo một máy tính, bạn không thể viết mã để cộng, trừ, nhân, chia nhiều lần.

#### Hoặc, nếu chương trình của bạn cần thêm 10 cặp chữ số khác nhau, bạn nên sử dụng một hàm cho việc này, thay vì viết mã 10 lần cho các giá trị khác nhau.

### **Moral**

#### Do đó, đạo đức của câu chuyện là, các chức năng cho phép chúng ta nhóm các nhiệm vụ lại với nhau và sử dụng chúng ở những nơi chúng ta yêu cầu, thay vì viết cùng một mã lặp đi lặp lại.

### **What are functions?**

#### Một **hàm** là một nhóm các câu lệnh cùng nhau thực hiện một tác vụ cụ thể.

#### Mỗi chương trình C++ có ít nhất một hàm, đó là main().

#### Một hàm chỉ thực thi khi nó được gọi.

### **Main() function**

#### Ôi chao!

#### Chúng tôi đã đáp ứng chức năng chính trước đây!

#### Nhớ...

#### hàm main() là điểm vào của bất kỳ chương trình C++ nào.

#### Đó là thời điểm mà tại đó việc thực thi một chương trình được bắt đầu.

#### Khi một chương trình C++ được thực thi, điều khiển thực thi sẽ trực tiếp đến hàm main().

#### Mỗi chương trình C++ đều có hàm main().

### **Why use function?**

#### Biết bạn biết chức năng là gì...

#### Hãy xem tại sao sử dụng chúng.

#### Các hàm được sử dụng để **chia một mã lớn thành các mô-đun**, do đó chúng ta có thể dễ dàng gỡ lỗi và duy trì mã.

#### Ví dụ: nếu chúng ta viết một chương trình máy tính tại thời điểm đó, chúng ta có thể viết mọi logic trong một hàm riêng biệt (Đối với phép cộng sum (), để trừ sub ()).

#### Bất kỳ chức năng có thể được gọi nhiều lần.

### **Advantages of functions**

* Khả năng tái sử dụng mã
* Phát triển một ứng dụng ở định dạng mô-đun.
* Dễ dàng gỡ lỗi chương trình.
* Tối ưu hóa mã: Không cần phải viết nhiều mã.

#### Vì vậy, hãy tiếp tục và làm việc với các hàm trong C ++

### **Structure of Function**

### **Getting to know function**

#### Khai báo hàm cho trình biên dịch biết về tên, kiểu trả về và tham số của hàm.

#### Một định nghĩa hàm cung cấp phần thân thực tế của hàm.

#### Một hàm được biết đến với nhiều tên khác nhau như một phương thức hoặc một chương trình con hoặc một thủ tục, v.v.

#### Thư viện chuẩn C++ cung cấp nhiều hàm tích hợp mà chương trình của bạn có thể gọi.

#### Ví dụ, cin, cout và nhiều hơn nữa.

#### Đây là những chức năng đã có sẵn trong hệ thống và chúng tôi chưa xác định chúng.

### **General Form**

#### Dạng chung của định nghĩa hàm C++ như sau:

### 

### 

#### Định nghĩa hàm C++ bao gồm tiêu đề hàm và nội dung hàm. Dưới đây là tất cả các phần của một hàm −

* Return Type − Một hàm có thể trả về một giá trị.

#### return\_type là kiểu dữ liệu của giá trị mà hàm trả về. Một số hàm thực hiện các thao tác mong muốn mà không trả về giá trị.

#### Trong trường hợp này, return\_type là khoảng trống từ khóa.

* Tên hàm - Đây là tên thực của hàm.

#### Tên hàm và danh sách tham số cùng nhau tạo thành chữ ký hàm.

* Tham số − Một tham số giống như một trình giữ chỗ.

#### Khi một hàm được gọi, bạn truyền một giá trị vào tham số.

#### Giá trị này được gọi là tham số hoặc đối số thực tế.

#### Danh sách tham số đề cập đến loại, thứ tự và số lượng tham số của hàm.

#### Các thông số là tùy chọn; nghĩa là, một hàm có thể không chứa tham số.

* Thân hàm - Thân hàm chứa một tập hợp các câu lệnh xác định chức năng của hàm.

### **Basic Function Example**

### **Implementation**

#### Ok, vì vậy hãy thực hiện những gì chúng ta đã học và tạo một hàm để tìm số nào lớn hơn trong số hai.

### **Declaring and Defining a function**

#### Sau đây là mã nguồn cho một hàm được gọi là greater().

#### Hàm này lấy hai tham số num1 và num2 và trả về số lượng lớn hơn của cả hai −

int greater(int num1, int num2) {

// local variable declaration

int result;

if (num1 > num2){

result = num1;

}

else{

result = num2;

}

return result;

}



#### Vì vậy, chúng ta hãy cố gắng hiểu hàm mà chúng ta vừa định nghĩa...

int greater(int num1, int num2)

* Phía trên dòng mã khai báo một hàm có tên lớn hơn, vì nó sẽ trả về giá trị số nguyên (số lớn hơn), kiểu trả về sẽ là int.
* Tiếp theo, nó chấp nhận hai tham số, num1 và num2 (Hai số mà chúng ta muốn tìm số lớn hơn)

### **Logic**

int result;

if (num1 > num2){

result = num1;

}

else{

result = num2;

}

return result;

#### Trên đây là logic để tìm số lượng lớn hơn

### **Half done**

#### Đã đi được một nửa!

#### Bây giờ chúng tôi đã khai báo chức năng...

#### Nhưng đến bây giờ, nó không hoạt động!

#### Tại sao?

#### Bởi vì - Một hàm chỉ thực thi khi nó được gọi.

### **Calling the function**

#### Trong khi tạo hàm C++, bạn đưa ra định nghĩa về những gì hàm phải làm.

#### Để sử dụng một hàm, bạn sẽ phải gọi hoặc gọi hàm đó.

#### Khi một chương trình gọi một hàm, điều khiển chương trình được chuyển sang hàm được gọi.

#### Một hàm được gọi thực hiện một tác vụ được xác định và khi câu lệnh trả về của nó được thực thi hoặc khi đạt đến dấu ngoặc nhọn đóng kết thúc hàm của nó, nó sẽ trả lại điều khiển chương trình trở lại chương trình chính.

#### Để gọi một hàm, bạn chỉ cần truyền các tham số cần thiết cùng với tên hàm và nếu hàm trả về một giá trị, thì bạn có thể lưu trữ giá trị trả về.

### **Almost done**

int main () {

// local variable declaration:

int a = 100;

int b = 200;

int ret;

// calling a function to get the greater value.

ret = max(a, b);

cout << "Greater value amongst the two is : " << ret << endl;

return 0;

}

Greater value amongst the two is : 200



#### Đó là cách các hàm hoạt động trong C++

### **Revisiting Strings**

### **You know them: Strings**

#### Ê

#### Hiện tại, bạn đã biết Strings là gì...

#### Hãy sửa đổi một chút.

#### Thuật ngữ chuỗi thường có nghĩa là một chuỗi các ký tự có thứ tự, với ký tự đầu tiên, ký tự thứ hai, v.v. và trong hầu hết các ngôn ngữ lập trình, các chuỗi như vậy được đặt trong dấu ngoặc kép đơn hoặc kép.

#### Trong C++, dấu phân cách kèm theo là dấu ngoặc kép.

#### Một chuỗi đại diện cho một chuỗi các ký tự.

### **Two ways**

#### C++ cung cấp hai loại biểu diễn chuỗi sau:

* Chuỗi ký tự kiểu C
* Kiểu lớp chuỗi được giới thiệu với Standard C++.

### **The C-Style Character String**

#### Vì C ++ bắt nguồn rất nhiều từ C, chuỗi ký tự kiểu C có nguồn gốc từ ngôn ngữ C và tiếp tục được hỗ trợ trong C ++.

#### Chuỗi này thực sự là một mảng ký tự một chiều được chấm dứt bởi một ký tự null '\0'.

#### Do đó, một chuỗi kết thúc null chứa các ký tự bao gồm chuỗi theo sau là null.

### **Example**

#### Khai báo và khởi tạo sau đây tạo một chuỗi bao gồm từ "Xin chào".

#### Để giữ ký tự null ở cuối mảng, kích thước của mảng ký tự chứa chuỗi nhiều hơn một so với số ký tự trong từ "Xin chào".

char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

#### Nếu bạn tuân theo quy tắc khởi tạo mảng, thì bạn có thể viết câu lệnh trên như sau:

char greeting[] = "Hello";

### **The String Class in C++**

#### Thư viện C++ tiêu chuẩn cung cấp một kiểu lớp chuỗi hỗ trợ các hoạt động khác nhau và bổ sung nhiều chức năng hơn.

#### Hãy để chúng tôi kiểm tra ví dụ sau:

string name = ”John”;

cout << name;

#### GỢI Ý: Chúng ta đã thấy điều này trong các chủ đề trước

#### String thực sự là một lớp trong thư viện chuẩn C ++.

#### Nó được định nghĩa trong thư viện tiêu chuẩn <**chuỗi> hoặc <string.h>.**

#### Vì vậy, khi bạn khai báo một đối tượng của **chuỗi**, về cơ bản bạn đang khởi tạo một đối tượng của chuỗi lớp.

#### Đừng lo lắng nếu bạn không biết về các khái niệm như lớp học và hơn thế nữa, tôi đã giúp bạn được đề cập trong khóa học C ++ Advanced.

### **String Functions**

#### Chúng ta biết, một chuỗi là một lớp trong C ++.

#### Do đó, nó cũng chứa một số chức năng được xác định trước trong C ++ Hãy kiểm tra chúng!

### **String functions**

1. strcpy(s1, s2): Sao chép chuỗi s2 vào chuỗi s1
2. strcat(s1, s2): Nối chuỗi s2 vào cuối chuỗi s1
3. strlen(s1): Trả về độ dài của chuỗi s1
4. strcmp (s1, s2): Trả về 0 nếu s1 và s2 giống nhau; nhỏ hơn 0 nếu s1<s2; lớn hơn 0 nếu S1>S2
5. strchr(s1, ch): Trả về một con trỏ đến lần xuất hiện đầu tiên của ký tự ch trong chuỗi s1
6. strstr(s1, s2): Trả về một con trỏ đến lần xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1

### **Example**

#include <iostream>

#include <string>

using namespace std;

int main () {

string str1 = "Hello";

string str2 = "World";

string str3;

int len ;

// copy str1 into str3

strcpy( str3, str1);

cout << "strcpy( str3, str1) : " << str3 << endl;

// concatenates str1 and str2

strcat( str1, str2);

cout << "strcat( str1, str2): " << str1 << endl;

// total length of str1 after concatenation

len = strlen(str1);

cout << "strlen(str1) : " << len << endl;

return 0;

}

strcpy( str3, str1) : Hello strcat( str1, str2): HelloWorld strlen(str1) : 10

#### Khá đơn giản, phải không?

#### Này bạn đời! Chúng tôi gần như đã hoàn thành...

#### Trước khi tôi ký tên, hãy để tôi đưa bạn qua một bản sửa đổi nhỏ về chủ đề này như mọi khi...

### **The End**

#### Ê! Chúc mừng bạn đã hoàn thành khóa học!

#### Bạn là một học sinh giỏi và thật tuyệt khi dạy bạn viết mã... Cảm ơn bạn!

#### MẸO: Tiếp tục luyện tập, Tiếp tục viết mã



#### Cũng xin chúc mừng bạn đã đạt được chứng chỉ của mình, bằng cách hoàn thành khóa học này.

#### Bạn đã làm việc chăm chỉ cho việc này, vì vậy hãy phô trương trước mặt bạn bè và trên phương tiện truyền thông xã hội và đưa chứng chỉ này cho bất cứ ai có thể ngạc nhiên!

#### Đó là món quà của sự kiên nhẫn và chăm chỉ của bạn!



#### Vì vậy, đó là nó từ bên này!

#### Bản thân tôi, Bjarne Stroustrup, ký tên!

#### Hy vọng sẽ gặp lại bạn trong khóa học Nâng cao... Taddaaa!