Bài 24: GIỚI THIỆU VỀ CẤU TRÚC ĐIỀU KHIỂN (CONTROL FLOW INTRODUCTION)

Xem bài học trên website để ủng hộ Kteam: Giới thiệu về cấu trúc điều khiển (Control flow introduction)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở bài học trước, bạn đã nắm khái niệm TIỀN KHAI BÁO VÀ ĐỊNH NGHĨA HÀM (Forward declarations and Definitions of Functions).

Hôm nay, mình sẽ giúp các bạn nắm được tổng quan về luồng xử lý của một chương trình C++ qua bài: **Giới thiệu về cấu trúc điều khiển (Control flow introduction)**.

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- CẤU TRÚC MỘT CHƯƠNG TRÌNH C++ (Structure of a program)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Tổng quan về cấu trúc điều khiển trong C++

- Câu lệnh dừng (halt)
- Câu lệnh nhảy (Jumps)
- Cấu trúc rẽ nhánh có điều kiện (Conditional branches)
- Cấu trúc vòng lặp (Loops)
- Xử lý ngoại lệ (Exceptions handling)

Tổng quan về cấu trúc điều khiển trong C++

Như bạn đã biết, khi một chương trình C++ được chạy, CPU sẽ bắt đầu thực thi các câu lệnh ở **vị trí đầu hàm main()**, thực thi các câu lệnh **tuần tự từ trên xuống dưới**, **kết thúc chương trình ở cuối hàm main()**.

Trong thực tế, nếu thực thi một cách tuần tự như vậy, sẽ có rất nhiều vấn đề không thể giải quyết.

Ví dụ về chương trình thực hiện một chức năng nào đó dựa trên dữ liệu của người dùng, hoặc chương trình yêu cầu thực hiện lặp lại một công việc nào đó. Lúc này, chúng ta cần can thiệp và thay đổi luồng xử lý của chương trình

Vì vậy, ngôn ngữ C++ cung cấp các **câu lệnh điều khiển (control flow statements)** cho phép lập trình viên **thay đổi luồng xử lý** của chương trình. Bài học này, mình sẽ giới thiệu một cách tổng quan về các câu lệnh điều khiển trong C++.

Câu lệnh dừng (halt)

Câu lệnh điều khiển dừng (halt) là một cấu trúc **cơ bản nhất**, nó yêu cầu chương trình **ngừng làm việc ngay lập tức**.

Trong C++, câu lệnh dừng được thực hiện thông qua hàm **exit()** trong thư viện **cstdlib**. Hàm **exit()** nhận vào một số nguyên và nó sẽ được trả về cho hệ điều hành như một mã kết thúc chương trình, tương tự như giá trị trả về của hàm main.

Ví dụ chương trình sử dụng hàm exit():

```
#include <cstdlib> // needed for exit()
#include <iostream>
using namespace std;

int main()
{
	cout << 1;
	exit(0); // terminate and return 0 to operating system

	// The following statements never execute
	cout << 2;
	return 0;
}
```

Khi gặp câu lệnh ***exit*(0);** chương trình sẽ dừng ngay lập tức và trả về giá trị 0 cho hệ điều hành.

Câu lệnh nhảy (Jumps)

Câu lệnh nhảy (jumps) giúp CPU **nhảy đến thực thi một dòng lệnh khác**. Các từ khóa sử dụng cho cấu trúc nhảy: **goto**, **break**, **continue**. Những từ khóa này sẽ được giới thiệu trong những bài học tiếp theo.

Lời gọi hàm cũng hoạt động như một câu lệnh nhảy. Khi một hàm được gọi, CPU nhảy lên câu lệnh đầu tiên của hàm được gọi. Khi hàm được gọi kết thúc, CPU quay lại câu lệnh sau lời gọi hàm.

Cấu trúc rẽ nhánh có điều kiện (Conditional branches)

Cấu trúc rẽ nhánh có điều kiện (Conditional branches) làm chương trình **thay đổi hướng thực thi** dựa trên giá trị của biểu thức điều kiện (hoặc các mệnh đề).

Tiêu biểu cho cấu trúc rẽ nhánh là câu lệnh if:

```
int main()
{
	// do A
	if (expression)
		// do B
	else
		// do C

	// do D
}
```

Nếu **expression** là đúng (true), thứ tự thực thi của chương trình sẽ là **A – B – D**. Nếu **expression** là sai (false), thứ tự sẽ là **A – C – D**.

Từ khóa **switch...case...** cũng là một cấu trúc rẽ nhánh có điều kiện.

Cấu trúc vòng lặp (Loops)

Cấu trúc vòng lặp (loops) giúp chương trình thực hiện lặp lại một khối lệnh, đến khi không còn thỏa mãn điều kiện lặp.

```
int main()
{
	//do A
	//do B 0 or more times
	//do C
}
```

Chương trình trên có thể thực hiện theo hướng ABC, ABBC, ABBBC, ABBB...BBBC, hoặc cũng có thể là AC, tùy vào điều kiện lặp của chương trình.

Ngôn ngữ C++ cung cấp 4 loại vòng lặp: **while**, **do while**, **for** và **for each** (C++ 11). Mình sẽ nói rõ về nó trong những bài tiếp theo.

Xử lý ngoại lệ (Exceptions handling)

Ngoại lệ (exceptions) là một **cơ chế xử lý lỗi** xảy ra bên trong hàm. Nếu một lỗi xảy ra bên trong hàm, hàm đó sẽ ném ra một ngoại lệ (exception). Lúc này, chương trình sẽ **nhảy đến khối lệnh** chuyên dùng để xử lý ngoại lệ **có kiểu tương ứng** với ngoại lệ được hàm ném ra.

Exception Handling (xử lý ngoại lệ) trong C++ được xây dựng dựa trên 3 từ khóa là: **try**, **catch**, và **throw**. Nó là một tính năng cao cấp của ngôn ngữ C++, nên nó sẽ được đề cập ở những bài nâng cao về sau.

Kết luận

Qua bài học Giới thiệu về cấu trúc điều khiển (Control flow introduction), bạn đã nắm được tổng quan về luồng xử lý và các loại cấu trúc điều khiển trong C++.

Trong bài tiếp theo, mình sẽ giới thiệu về cấu trúc rẽ nhánh trong C++, cụ thể là CÂU ĐIỀU KIỆN IF & TOÁN TỬ ĐIỀU KIỆN TRONG C++ (If statements).

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".