

Bài 4: Vòng lặp While (While Loop)

CPP

BỨC TRANH TOÀN CẢNH: SỨC MẠNH TỰ ĐỘNG HÓA

Hãy tưởng tượng bạn đang xúc cát để lấp đầy một cái hồ.

- **Manual (Không vòng lặp):** Tự nhủ "Xúc một xẻng. Xúc thêm một xẻng...". Rất mệt và dễ quên đếm.
- **Automation (Có vòng lặp):** Ra lệnh cho Robot một lần duy nhất: "Này Robot, KHI MÀ (WHILE) cái hồ chưa đầy, hãy cứ xúc cát liên tục cho ta!"

Lộ trình khám phá (Roadmap)

Dựa trên cấu trúc bài học, chúng ta có 2 "khối" kiến thức (chunks) quan trọng:

- **Chương 1: "Người Gác Cổng" Nghiêm Khắc**
 - Cơ chế hoạt động Tại sao gọi là "Người gác cổng".
 - Phân tích cú pháp: `'while (condition) { code }'`.
- **Chương 2: Nghệ thuật Đếm lùi (Ứng dụng thực tế)**
 - Ví dụ "Countdown" (Đếm ngược).
 - Do/While Loop: Sự khác biệt với While.
 - Những cạm bẫy cần tránh (Vòng lặp vô tận).

Chương 1: "Người Gác Cổng" Nghiêm Khắc

Tại sao gọi là "**Người Gác Cổng**"?

- Vòng lặp 'while' làm việc rất nguyên tắc: **Kiểm tra giấy tờ trước, cho vào sau.**
- Nếu ngay từ đầu điều kiện đã sai, cánh cổng sẽ không bao giờ mở ra.

Cú pháp

```
1 // Điều kiện được kiểm tra ĐẦU TIÊN
2 while (condition) {
3     // Khối lệnh này chỉ chạy khi condition là TRUE
4     // code to be executed
5 }
```

Chương 2: Nghệ thuật Đếm lùi (The Countdown)

Phép ẩn dụ: Đồng hồ đếm ngược đêm Giao thừa.

Hãy tưởng tượng bạn đang chờ đợi khoảnh khắc bắn pháo hoa:

Code mô phỏng

```
1 int countdown = 3; // Bắt đầu từ số 3
2
3 while (countdown > 0) {           // Khi nào vẫn còn thời gian (số > 0)
4     cout << countdown << "\n";   // Hết to số đó lên!
5     countdown--;                  // Thời gian trôi đi (Giảm 1 đơn vị)
6 }
7
8 cout << "Happy New Year!!\n"; // Pháo hoa nổ tung!
```

Dòng chữ "Happy New Year!!" nằm *ngoài* vòng lặp nên nó được chạy ngay khi vòng lặp kết thúc.

Phân tích chuyển động: Đi xuống cầu thang

Ở đây chúng ta dùng 'countdown-' thay vì 'i++'.

- **'-' (Decrement):** Hãy hình dung nó giống như bạn đang đi xuống cầu thang.
- Đang ở bậc 3 → Hết "3" → Bước xuống (còn 2).
- Đang ở bậc 2 → Hết "2" → Bước xuống (còn 1).
- Đang ở bậc 1 → Hết "1" → Bước xuống (còn 0).
- **Tại bậc 0:** Người gác cổng nhìn thấy số 0. Điều kiện ' $0 > 0$ ' là **SAI**. Cổng đóng lại.

Mở rộng: Vòng lặp "Làm trước, Nghĩ sau"(Do/While)

Nếu 'While' là "Người Gác Cổng" cản trọng, thì '**Do/While**' là một "**Gã Liều Lĩnh**".

Triết lý của Do/While

"Cứ làm thử một lần đi đã, rồi tính tiếp!"

Cú pháp Do/While

```
1 do {  
2     // Làm gì đó đi! (Code block)  
3 }  
4 while (condition);
```

Sự khác biệt cốt tử:

- **While:** Kiểm tra vé **trước khi** vào cổng.
- **Do/While:** Cho phép vào chơi **ít nhất một lần**, sau đó mới kiểm tra vé ở cổng ra.

Kiểm tra sự thấu hiểu (Check-in)

Hãy giải câu đố sau để phân biệt hai loại vòng lặp:

Câu đố

Giả sử tôi có biến `int i = 0`.

❶ **Trường hợp A:** Dùng `while (i > 0) { cout << "Hello"; }`

❷ **Trường hợp B:** Dùng `do { cout << "Hello"; } while (i > 0);`

Trong mỗi trường hợp, chữ "Hello" được in ra màn hình mấy lần?

Kiểm tra sự thấu hiểu (Check-in)

Hãy giải câu đố sau để phân biệt hai loại vòng lặp:

Câu đố

Giả sử tôi có biến `'int i = 0'`.

❶ **Trường hợp A:** Dùng `'while (i > 0) { cout << "Hello"; }'`

❷ **Trường hợp B:** Dùng `'do { cout << "Hello"; } while (i > 0);'`

Trong mỗi trường hợp, chữ "Hello" được in ra màn hình mấy lần?

Đáp án:

- **Trường hợp A (While):** "Người gác cổng" chặn ngay từ đầu ($0 > 0$ sai). **0 lần.**
- **Trường hợp B (Do/While):** "Gã liều lĩnh" cho làm trước 1 lần rồi mới kiểm tra. **1 lần.**

Mảnh ghép cuối cùng: Ứng dụng thực tế

Khi nào dùng cái nào? Hãy học tư duy giải quyết vấn đề.

1. Do/While → Nhập mật khẩu

Bạn luôn muốn người dùng nhập *ít nhất một lần*.

```
1 int password;  
2 do {  
3     cout << "Moi nhap pass: ";  
4     cin >> password;  
5 }  
6 while (password != 1234);  
7 // Sai (khác 1234) thì nhập lại!
```

Ẩn dụ: Ăn thử một miếng rồi mới quyết định ăn tiếp hay không.

2. While → Kiểm tra bình xăng

Trước khi nổ máy, phải kiểm tra xăng.

- Xăng hết → Không đi mét nào (0 lần).
- Còn xăng → Lái xe, xăng với dần.

Ẩn dụ: An toàn là trên hết. Kiểm tra trước khi hành động.

Tổng kết hành trình

Chúng ta đã hoàn thành xuất sắc bài học về **Vòng lặp While!**

- ➊ **While Loop:** Người gác cổng nghiêm khắc. Kiểm tra trước, làm sau. (Dừng khi có thể không cần làm lần nào).
- ➋ **Do/While Loop:** Gã liều lĩnh. Làm trước, kiểm tra sau. (Dừng khi bắt buộc phải làm ít nhất 1 lần).
- ➌ **Vòng lặp vô tận:** Cơ ác mòng khi quên thay đổi biến đếm (quên 'i++' hoặc 'i--').

Bước tiếp theo

Thông thường, sau khi học 'While' (không biết trước số lần), chúng ta sẽ chuyển sang '**For Loop**'. Bạn có muốn tiếp tục sang bài For Loop ngay không?