

Bí Mật Của Đoàn Tàu Ký Tự

C++ String & Iterator

Learning How to Learn

Ngày 14 tháng 1 năm 2026

Mục Lục: Lộ Trình Khám Phá

Chương 1: Đoàn tàu và Người dẫn đường

Hiểu bản chất string là đoàn tàu và iterator là người soát vé.

Chương 2: Hai cách để đi hết một con đường

Duyệt qua đoàn tàu bằng vòng lặp while (Dò đường) và for (Lập trình sẵn).

Chương 3: Phép thuật của Thầy phù thủy

Sử dụng Algorithms (sort, reverse) để thay đổi trật tự đoàn tàu.

Chương 1: Đoàn Tàu và Người Dẫn Đường

Phép ẩn dụ: Đoàn tàu "String"

Hãy tưởng tượng từ **"HELLO"** là một đoàn tàu gồm 5 toa xe:

- Mỗi toa chở một hành khách: H - E - L - L - O.
- Cả đoàn tàu chính là biến `string`.

Nhân vật chính: Iterator (Người Soát Vé)

- Không phải là toa tàu, không phải hành khách.
- Là người **đang đứng tại** một toa và chỉ tay vào hành khách.
- Trong C++ gọi là: `string::iterator`.

Hai nhà ga quan trọng: `.begin()` và `.end()`

Để Người Soát Vé làm việc, họ cần biết điểm đầu và điểm cuối.

- `.begin()`: Người Soát Vé đứng ngay tại toa đầu tiên (chữ 'H').

Cảnh báo quan trọng về `.end()`

`.end()` (Vực thăm): Đây là chỗ dễ nhầm lẫn nhất!

- `.end()` **KHÔNG PHẢI** là toa cuối cùng (chữ 'O').
- Nó là mặt đất **ngay sau** toa cuối cùng.
- Nếu đi đến `.end()`, nghĩa là đã bước ra khỏi tàu (Hết tàu!).

Code minh họa: Triệu hồi Người Soát Vé

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     // 1. Tao ra doan tau
7     string doan_tau = "HELLO";
8
9     // 2. Thue mot Ngươi Soat Ve (khai bao iterator)
10    string::iterator nguoi_soat_ve;
11
12    // 3. Cho nguoi soat ve dung o toa dau tien
13    nguoi_soat_ve = doan_tau.begin();
14
15    // 4. Hoi nguoi soat ve: "Ông đang thay ai?"
16    // Dau * giống như đôi mắt nhìn vào toa tau
17    cout << "Hành khách đầu tiên: " << *nguoi_soat_ve << endl;
18
19    return 0;
20 }
```

*Lưu ý: Dấu * dùng để truy cập giá trị (hành khách) tại vị trí iterator đang đứng.*

Trạm dừng kiểm tra (Checkpoint 1)

Câu hỏi

Nếu tôi ra lệnh cho `nguo_i_soat_ve` đi đến vị trí `doan_tau.end()`, và tôi cố tình dùng dấu `*` để hỏi xem ông ấy thấy gì (`*nguo_i_soat_ve`), chuyện gì sẽ xảy ra?

- A. Ông ấy thấy chữ cái cuối cùng của đoàn tàu (chữ 'O').
- B. Ông ấy thấy toa tàu đầu tiên.
- C. Lỗi! Ông ấy đang đứng dưới đường ray (ngoài đoàn tàu) nên không thấy ai cả.

Trạm dừng kiểm tra (Checkpoint 1)

Câu hỏi

Nếu tôi ra lệnh cho `nguo_i_soat_ve` đi đến vị trí `doan_tau.end()`, và tôi cố tình dùng dấu `*` để hỏi xem ông ấy thấy gì (`*nguo_i_soat_ve`), chuyện gì sẽ xảy ra?

- A. Ông ấy thấy chữ cái cuối cùng của đoàn tàu (chữ 'O').
- B. Ông ấy thấy toa tàu đầu tiên.
- C. Lỗi! Ông ấy đang đứng dưới đường ray (ngoài đoàn tàu) nên không thấy ai cả.

Đáp án: C. Chính xác! `.end()` là vực thẳm, cố nhìn vào đó sẽ gây lỗi chương trình (crash).

Chương 2: Vũ khí bí mật AUTO

Thay vì viết chức danh dài dòng: `string::iterator` `nguai_soat_ve`, chúng ta dùng `auto`.

Tại sao dùng `auto`?

`auto` giống như bộ đồ tắc kè hoa. Máy tính tự hiểu: "Ông này đứng trên tàu `String`, nên chắc chắn là `String Iterator`".

Code cũ:

```
string::iterator nguoi_soat_ve = doan_tau.begin();
```

Code mới (Gọn hơn):

```
auto nguoi_soat_ve = doan_tau.begin();
```


Cách 1: Phong cách "Dò Đường"(WHILE Loop)

Giống như đi bộ cẩn thận. Vừa đi vừa tự hỏi: "Mình đã rơi khỏi tàu chưa?".

```
1  string doan_tau = "CODING";
2  auto it = doan_tau.begin();
3
4  cout << "Duyet bang WHILE: ";
5
6  // "Chung nao toi chua cham den vuc tham (.end)"
7  while (it != doan_tau.end()) {
8      // Lam viec: In hanh khach ra
9      cout << *it << " ";
10
11     // QUAN TRONG: Buoc sang toa tiep theo
12     it++;
13 }
```

Lưu ý: `it++` là bước chân. Nếu quên, iterator đứng mãi một chỗ (vòng lặp vô tận).

Cách 2: Phong cách "Lập trình sẵn"(FOR Loop)

Giống như đi trên băng chuyền. Gom tất cả quy tắc vào một dòng.

- ❶ **Khởi động:** `auto it = doan_tau.begin()`
- ❷ **Điều kiện:** `it != doan_tau.end()`
- ❸ **Bước nhảy:** `it++`

```
1  cout << "Duyet bang FOR: ";
2
3  // Gom tat ca vao mot dong
4  for (auto it = doan_tau.begin(); it != doan_tau.end(); it++) {
5      cout << *it << " ";
6  }
```

WHILE (Đi bộ tự do)

Bạn phải tự nhớ để bước đi (`it++`) ở bên trong. Linh hoạt nhưng dễ quên bước chân.

FOR (Băng chuyền tự động)

Mọi thứ (xuất phát, đích đến, bước đi) đều được cài đặt ngay từ đầu. An toàn và khó quên hơn.

Trạm dừng kiểm tra (Checkpoint 2)

Câu hỏi tình huống

Trong vòng lặp `while`, nếu lỡ tay viết nhầm: Cho `it++` (bước đi) lên **trước** câu lệnh `cout << *it` (in ra). Chuyện gì xảy ra?

- A. Mọi thứ vẫn bình thường.
- B. Bỏ qua toa đầu, và khi đến cuối sẽ lỗi vì bước ra ngoài vực thăm rồi mới nhìn.
- C. Nó sẽ in ngược từ dưới lên.

Trạm dừng kiểm tra (Checkpoint 2)

Câu hỏi tình huống

Trong vòng lặp `while`, nếu lỡ tay viết nhầm: Cho `it++` (bước đi) lên **trước** câu lệnh `cout << *it` (in ra). Chuyện gì xảy ra?

- A. Mọi thứ vẫn bình thường.
- B. Bỏ qua toa đầu, và khi đến cuối sẽ lỗi vì bước ra ngoài vực thăm rồi mới nhìn.
- C. Nó sẽ in ngược từ dưới lên.

Đáp án: B.

- **Bỏ sót:** Bước đi rồi mới nhìn -> Toa đầu bị bỏ qua.
- **Tai nạn:** Bước ra `.end()` rồi mới nhìn -> Lỗi chương trình.

Chương 3: Phép thuật của Thầy phù thủy (Algorithms)

Cần thư viện: `#include <algorithm>`

Nguyên lý hoạt động: Luôn hỏi 2 câu:

- 1 "Bắt đầu làm phép từ đâu?"(`.begin()`)
- 2 "Dừng lại trước chỗ nào?"(`.end()`)

Hai phép thuật phổ biến

1. Sắp xếp (Sort): Biến đoàn tàu lộn xộn thành ngăn nắp (A-Z).

`sort(s.begin(), s.end());`

2. Đảo ngược (Reverse): Quay đầu toàn bộ đoàn tàu.

`reverse(s.begin(), s.end());`

Code minh họa Algorithms

```
1 #include <iostream>
2 #include <string>
3 #include <algorithm> // Bat buoc de dung phep thuat
4 using namespace std;
5
6 int main() {
7     string s = "PYTHON";
8     cout << "Ban dau: " << s << endl;
9
10    // 1. Phep thuat Sap Xep (Sort)
11    sort(s.begin(), s.end());
12    cout << "Sau khi sort: " << s << endl;
13    // Ket qua: HNOPTY
14
15    // 2. Phep thuat Dao Nguoc (Reverse)
16    reverse(s.begin(), s.end());
17    cout << "Sau khi reverse: " << s << endl;
18    // Ket qua: YTPONH
19
20    return 0;
21 }
```

Thử thách tốt nghiệp (Final Boss)

Đề bài

Có chuỗi string `s = "hocbai";`.

Muốn sắp xếp **chỉ 3 chữ cái đầu** ("hoc") thành "cho", giữ nguyên "bai".

Kết quả: "chobai".

Chọn câu lệnh đúng:

- A. `sort(s.begin(), s.end());`
- B. `sort(s.begin(), s.begin() + 3);`
- C. `sort(s.begin() + 3, s.end());`

Thử thách tốt nghiệp (Final Boss)

Đề bài

Có chuỗi string `s = "hocbai";`.

Muốn sắp xếp **chỉ 3 chữ cái đầu** ("hoc") thành "cho", giữ nguyên "bai".

Kết quả: "chobai".

Chọn câu lệnh đúng:

- A. `sort(s.begin(), s.end());`
- B. `sort(s.begin(), s.begin() + 3);`
- C. `sort(s.begin() + 3, s.end());`

Đáp án: B.

- Vị trí kết thúc luôn là vị trí đứng **ngay sau** phần tử muốn tác động.
- Muốn xếp 3 toa đầu -> Cần chỉ vào vạch ngăn cách sau toa thứ 3 (`begin() + 3`).

Lễ Tốt Nghiệp: Tổng Kết

Chúc mừng bạn đã làm chủ **Đoàn Tàu Ký Tự**!

Hành trang đã thu thập

- ❶ **Iterator**: Ngón tay chỉ vị trí (Người soát vé).
- ❷ **Loop (For/While)**: Hai phong cách đi bộ trên tàu.
- ❸ **Algorithms (Sort/Reverse)**: Phép thuật dựa trên điểm đầu và điểm cuối.

Bước tiếp theo?

- Thực chiến với bài toán mini?
- Khám phá **Vector** (Đoàn tàu vô tận)?