Bí Kíp Luyện Rồng C++: Vector & Iterator

Khám phá Chiếc Túi Thần Kỳ và Người Soát Vé

Slide Learning Cpp

Ngày 14 tháng 1 năm 2026

Bản Đồ Kho Báu C++ (Lộ trình học tập)

Hành trình 4 Chương

- **Chương 1: Vector - Đoàn Tàu Co Giãn:** Tại sao cần Vector thay vì mảng thường?
- **Chương 2: Iterator - Người Soát Vé Tận Tụy:** Hiểu về con trỏ thông minh.
- **Chương 3: Hành Trình Điểm Danh:** Duyệt Vector bằng vòng lặp.
- **Chương 4: Phép Thuật Algorithms:** Sắp xếp và Tìm kiếm.

Chương 1: Vector - Đoàn Tàu Co Giãn

Mảng (Array) Cũ Kỹ

Giống như khay đựng trứng nhựa cứng.

- Cố định số lượng (VD: 6 lỗ).
- Muốn thêm quả thứ 7? Phải mua khay mới!

Vector (Mảng Động)

Giống như Đoàn Tàu Phép Thuật.

- Tự động thêm toa khi có khách mới.
- Có thể dài ra vô tận.

Cốt lõi: Vector là danh sách có thể **tự động co giãn** kích thước.

Câu thần chú: Triệu hồi Vector

1. **Mở sách phép (Thư viện):**

```cpp
#include <vector>
using namespace std;
```

2. **Tạo đoàn tàu (Khai báo):**

```cpp
vector<string> vu_khi; // Doan tau rong
```

3. **Nối thêm toa (Push Back):** Đẩy món đồ vào cửa sau (‘back‘) của đoàn tàu.

```cpp
vu_khi.push_back("Kiem Go");    // Toa 1
vu_khi.push_back("Cung Ten");   // Toa 2 noi sau
vu_khi.push_back("Khien Sat"); // Toa 3 noi tiep
```

Kiểm tra nhanh (Quick Check)

Tình huống

Đoàn tàu `vu_khi` đang có 3 món: Kiếm, Cung, Khiên. Bạn hô thần chú:

```
1 vu_khi.push_back("Mu Giap");
2
```

Câu hỏi:

1. Món "Mũ Giáp"nằm ở vị trí nào? (Đầu, Giữa hay Cuối?)
2. Tổng cộng đoàn tàu có bao nhiêu toa?

Kiểm tra nhanh (Quick Check)

Tình huống

Đoàn tàu `vu_khi` đang có 3 món: Kiếm, Cung, Khiên. Bạn hô thần chú:

```
1 vu_khi.push_back("Mu Giap");
2
```

Câu hỏi:

1. Món "Mũ Giáp"nằm ở vị trí nào? (Đầu, Giữa hay Cuối?)
2. Tổng cộng đoàn tàu có bao nhiêu toa?

Đáp án

1. Người mới đến luôn xếp hàng ở **Cuối tàu**.
2. Tổng cộng có **4 toa**.

Chương 2: Iterator - Người Soát Vé Tận Tụy

Tại sao không dùng số thứ tự (Index) `vu_khi[0]`?

- Index giống như dịch chuyển tức thời (nhanh nhưng dễ nhầm chỗ).
- **Iterator** là Người Soát Vé đi bộ từng bước.

Cách hoạt động

- Bắt đầu từ đầu tàu.
- Bước từng bước sang toa kế (‘++‘).
- Biết chính xác khi nào hết đường ray (‘end()‘).

Ba hành động của Người Soát Vé

Khai báo: `vector<string>::iterator it;`

1. **Bắt đầu hành trình (begin):**

```
it = vu_khi.begin(); // Dung o cua toa dau tien
```

2. **Mở cửa kiểm tra (*it):** Dùng dấu sao * để xem bên trong.

```
cout << *it; // Hien ra "Kiem Go"
```

3. **Bước sang toa kế (it++):**

```
it++; // Buoc sang toa "Cung Ten"
```

Bí ẩn về điểm kết thúc (end)

Cảnh báo quan trọng!

`vu_khi.end()` **KHÔNG PHẢI** là toa cuối cùng.

Hãy tưởng tượng:

- `begin()`: Bậc thềm toa 1.
- `end()`: **Mặt đất sân ga**, ngay sau khi bước xuống khỏi toa cuối.

Khi `it == end()` nghĩa là "Chân đã chạm đất", đã đi hết tàu.

Kiểm tra nhanh (Quick Check)

Đoàn tàu: [Toa 1: Kiếm] [Toa 2: Cung] [Toa 3: Khiên] [Toa 4: Mũ]

```
vector<string>::iterator it = vu_khi.begin(); // (1)
it++; // (2) Buoc 1
it++; // (3) Buoc 2
cout << *it; // (4) Mo cua
```

Câu hỏi: Người soát vé đang đứng trước món đồ nào?

Kiểm tra nhanh (Quick Check)

Đoàn tàu: [Toa 1: Kiếm] [Toa 2: Cung] [Toa 3: Khiên] [Toa 4: Mũ]

```
vector<string>::iterator it = vu_khi.begin(); // (1)
it++; // (2) Buoc 1
it++; // (3) Buoc 2
cout << *it; // (4) Mo cua
```

Câu hỏi: Người soát vé đang đứng trước món đồ nào?

Đáp án

Sau 2 lần bước, ông ấy đứng ở **Toa 3: Khiên Sắt**.

Kiểm tra nhanh (Quick Check)

Đoàn tàu: [Toa 1: Kiếm] [Toa 2: Cung] [Toa 3: Khiên] [Toa 4: Mũ]

```
vector<string>::iterator it = vu_khi.begin(); // (1)
it++; // (2) Buoc 1
it++; // (3) Buoc 2
cout << *it; // (4) Mo cua
```

Câu hỏi: Người soát vé đang đứng trước món đồ nào?

Đáp án

Sau 2 lần bước, ông ấy đứng ở **Toa 3: Khiên Sắt**.

Câu hỏi phụ: Nếu bước thêm 2 bước nữa (`it++` 2 lần), ông ấy đứng ở đâu?

Kiểm tra nhanh (Quick Check)

Đoàn tàu: [Toa 1: Kiếm] [Toa 2: Cung] [Toa 3: Khiên] [Toa 4: Mũ]

```
vector<string>::iterator it = vu_khi.begin(); // (1)
it++; // (2) Buoc 1
it++; // (3) Buoc 2
cout << *it; // (4) Mo cua
```

Câu hỏi: Người soát vé đang đứng trước món đồ nào?

Đáp án

Sau 2 lần bước, ông ấy đứng ở **Toa 3: Khiên Sắt**.

Câu hỏi phụ: Nếu bước thêm 2 bước nữa (`it++` 2 lần), ông ấy đứng ở đâu? → Chạm đất (`end()`).

Duyệt Vector: Cách 1 (While)

Quy tắc: "Chừng nào chân ông chưa chạm đất (`!= end()`), thì cứ kiểm tra và đi tiếp."

```
vector<string>::iterator it;
it = vu_khi.begin();          // 1. Xuat phat

while (it != vu_khi.end()) { // 2. Kiem tra: Chua cham dat?
    cout << *it << endl;      // 3. Hanh dong: Doc ten
    it++;                     // 4. Di chuyen: Buoc tiep
}
```

Duyệt Vector: Cách 2 (For)

Gói gọn 3 bước vào 1 dòng code (Băng chuyền tự động).

```
// For (Khoi hanh ; Dieu kien dung ; Buoc di)
for (it = vu_khi.begin(); it != vu_khi.end(); it++) {
    cout << *it << endl;
}
```

- **Khởi hành:** `it = begin()`
- **Điều kiện dừng:** `it != end()` (Chưa chạm đất)
- **Bước đi:** `it++`

Thám tử tìm lỗi sai

Đoạn code sau có một lỗi sai kinh điển trong thế giới Iterator:

```
// Tim loi sai o dong nay:
for (it = vu_khi.begin(); it < vu_khi.end(); it++) {
    cout << *it;
}
```

Câu hỏi: Tại sao dùng dấu bé hơn < là sai, mà phải dùng !=?

Thám tử tìm lỗi sai

Đoạn code sau có một lỗi sai kinh điển trong thế giới Iterator:

```
// Tim loi sai o dong nay:
for (it = vu_khi.begin(); it < vu_khi.end(); it++) {
    cout << *it;
}
```

Câu hỏi: Tại sao dùng dấu bé hơn < là sai, mà phải dùng !=?

Giải thích

Trong C++, các toa tàu của một số loại container (như List, Map) nằm rải rác, không thẳng hàng nên không so sánh "bé hơn"được.
Chúng ta chỉ quan tâm: **"Đã đến đích chưa?"** (Khác đích hay bằng đích). Do đó luôn dùng !=.

Thay vì tự viết vòng lặp, hãy dùng thư viện `algorithm`.

Quy tắc chung

Phải chỉ rõ phạm vi tác động: Từ **Đầu tàu** đến **Cuối tàu**.

1. Phép thuật Sắp xếp (Sort):

```cpp
#include <algorithm>
// Sap xep tang dan (Be -> Lon, A -> Z)
sort(vu_khi.begin(), vu_khi.end());
```

Kết quả: {"Cung", "Khiên", "Kiếm"}

Phép thuật Tìm kiếm (Find)

Thả "Chó săn phép thuật"đi tìm đồ.

```cpp
// Cu phap: find(Bat dau, Ket thuc, Vat can tim);
auto ket_qua = find(vu_khi.begin(), vu_khi.end(), "Cung Ten");

if (ket_qua != vu_khi.end()) {
    cout << "Da tim thay!";
} else {
    cout << "Khong co trong balo!";
}
```

- **Tìm thấy:** `ket_qua` trỏ vào toa chứa đồ.
- **Không thấy:** `ket_qua` chạy thẳng ra `end()`.

Thử thách cuối cùng (The Final Boss)

Đề bài: Có danh sách điểm: `vector<int> diem = {7, 4, 9, 2};`
Nhiệm vụ:

1. Sắp xếp từ thấp đến cao.
2. In ra điểm thấp nhất và cao nhất.

Thử thách cuối cùng (The Final Boss)

Đề bài: Có danh sách điểm: `vector<int> diem = {7, 4, 9, 2};`
Nhiệm vụ:

1. Sắp xếp từ thấp đến cao.
2. In ra điểm thấp nhất và cao nhất.

Lời giải

```
sort(diem.begin(), diem.end()); // 1. Sap xep: {2, 4, 7, 9}

// 2. Diem thap nhat (Dau tau)
cout << "Min: " << diem[0];

// 3. Diem cao nhat (Cuoi tau - 1) HOAC dung back()
cout << "Max: " << diem.back();
```

Tổng Kết Khóa Học

Vũ Khí	**Ẩn Dụ**	**Tác Dụng**
Vector	Đoàn tàu co giãn	Mảng động, `push_back`
Iterator	Người soát vé	Con trỏ thông minh (`begin`, `end`)
For/While	Băng chuyền	Duyệt qua vector
Algorithm	Phép thuật	`sort`, `find`

Bạn đã sẵn sàng để viết code chưa?