Hành trình chinh phục C++: Queue và Deque

Slide Learning C++

2026

Lời chào

Chào bạn!

Tôi là người đồng hành của bạn trong hành trình chinh phục tri thức. Chúng ta sẽ cùng nhau khám phá thế giới lập trình C++, cụ thể là về **Queue (Hàng đợi)**.

1. BỨC TRANH TOÀN CẢNH (THE BIG PICTURE)

Hãy tưởng tượng bạn đang đứng trong một hàng đợi mua trà sữa:

- Người nào đến trước thì được mua trước và rời đi trước.
- Người nào đến sau thì phải đứng vào cuối hàng và đợi đến lượt.

Định nghĩa

Trong lập trình, Queue là một cấu trúc dữ liệu hoạt động theo nguyên tắc **FIFO (First In, First Out - Vào trước, Ra trước)**.

Lưu ý

Trong bài học này, chúng ta sẽ sử dụng `std::deque` để tận dụng sự linh hoạt và các công cụ hiện đại như `iterator`.

2. LỘ TRÌNH KHÁM PHÁ (THE MAP)

1. **Chương 1:** Khởi tạo Hàng Đợi.
2. **Chương 2:** Duyệt bằng `while`.
3. **Chương 3:** Vòng lặp `for` bán tự động.
4. **Chương 4:** `auto` Iterator.
5. **Chương 5:** Thư viện `<algorithm>`.

Sẵn sàng chưa?

Chúng ta sẽ bắt đầu ngay với Chương 1!

CHƯƠNG 1: KHỞI TẠO HÀNG ĐỢI

- `std::queue`: Giống ống nghiệm hẹp, chỉ thấy đầu và cuối.
- `std::deque`: Giống dãy hành lang kính, có thể nhìn thấy mọi vị trí.

Tại sao chọn deque?

Vì nó cho phép sử dụng `iterator` và `algorithms` một cách linh hoạt nhất.

Chương 1: Code mẫu

```cpp
#include <iostream>
#include <deque>
using namespace std;

int main() {
    // 1. Khoi tao hang doi rong
    deque<int> hang_doi;

    // 2. Them phan tu vao cuoi (Push Back)
    hang_doi.push_back(10); // Nguoi so 10 den truoc
    hang_doi.push_back(20);
    hang_doi.push_back(30);

    // 3. Xem nguoi dung dau (Front)
    cout << "Nguoi dung dau: " << hang_doi.front() << endl;

    return 0;
}
```

Kiểm tra nhanh (Check-point)

Câu hỏi

Nếu thực hiện lệnh:

1. `push_back(5)`
2. `push_back(15)`
3. `push_back(25)`

Lệnh `hang_doi.front()` sẽ trả về kết quả bao nhiêu? Tại sao?

Kiểm tra nhanh (Check-point)

Câu hỏi

Nếu thực hiện lệnh:

1. `push_back(5)`
2. `push_back(15)`
3. `push_back(25)`

Lệnh `hang_doi.front()` sẽ trả về kết quả bao nhiêu? Tại sao?

Đáp án

Con số **5** vì nó là người xếp hàng sớm nhất (FIFO).

CHƯƠNG 2: DUYỆT BẰNG WHILE

- **Quy trình:** Kiểm tra còn khách không → Phục vụ người đầu → Người đó rời đi.
- **Hành động:** Sử dụng `front()` và `pop_front()`.

Cảnh báo

Duyệt bằng `while` kết hợp `pop_front()` sẽ làm hàng đợi **trống rỗng** sau khi kết thúc.

Chương 2: Code mẫu

```cpp
#include <iostream>
#include <deque>
using namespace std;

int main() {
    deque<int> hang_doi = {5, 15, 25};
    while (!hang_doi.empty()) {
        int nguoi_dang_cho = hang_doi.front();
        cout << "Dang phuc vu: " << nguoi_dang_cho << endl;
        hang_doi.pop_front(); // Xoa khoi dau hang
    }
    return 0;
}
```

Kiểm tra sự hiểu biết

Câu hỏi

Nếu bạn quên dòng `hang_doi.pop_front();` trong vòng lặp `while`, chuyện gì sẽ xảy ra?

Kiểm tra sự hiểu biết

Câu hỏi

Nếu bạn quên dòng `hang_doi.pop_front();` trong vòng lặp `while`, chuyện gì sẽ xảy ra?

Đáp án

Lỗi **vòng lặp vô tận (infinite loop)**. Chương trình sẽ bị treo vì hàng đợi không bao giờ rỗng.

CHƯƠNG 3: VÒNG LẶP FOR BÁN TỰ ĐỘNG

- **Phép ẩn dụ:** Người quản lý đi điểm danh bằng mắt (Clipboard).
- **Đặc điểm:** Xem dữ liệu mà không làm mất dữ liệu.
- **Từ khóa** `auto`**:** Trợ lý thông minh tự nhận diện kiểu dữ liệu.

Chương 3: Code mẫu

```cpp
#include <iostream>
#include <deque>
#include <string>
using namespace std;

int main() {
    deque<string> hang_cho = {"An", "Binh", "Chi"};

    for (auto nguoi : hang_cho) {
        cout << "- " << nguoi << endl;
    }

    cout << "So nguoi van con: " << hang_cho.size() << endl;
    return 0;
}
```

CHƯƠNG 4: AUTO ITERATOR - CHIẾC KÍNH HIỂN VI

- **Iterator:** Giống như một con trỏ Laser.
- `begin()`: Trỏ vào người đầu tiên.
- `end()`: Trỏ vào vị trí **sau** người cuối cùng.
- `*it`: "Soi"giá trị tại vị trí laser đang trỏ.

Chương 4: Code mẫu

```
1     // Duyet bang Iterator
2     for (auto it = hang_cho.begin(); it != hang_cho.end(); ++it) {
3         cout << "Vi tri nay la: " << *it << endl;
4     }
```

Tư duy lập trình

Câu hỏi

Nếu muốn thay đổi tên tất cả khách hàng thành "Khách hàng thân thiết"tại chỗ, dùng `while` hay `iterator` hiệu quả hơn?

Tư duy lập trình

Câu hỏi

Nếu muốn thay đổi tên tất cả khách hàng thành "Khách hàng thân thiết"tại chỗ, dùng `while` hay `iterator` hiệu quả hơn?

Đáp án

Dùng **Iterator** hoặc **Range-based for (by reference)**. Nó cho phép sửa trực tiếp mà không cần xóa rồi thêm lại.

- `sort`: Sắp xếp hàng ngũ.
- `reverse`: Đảo ngược vị trí 180 độ.
- `find`: Tìm kiếm vị trí một người cụ thể.

Lưu ý

Các hàm này hoạt động dựa trên `iterator` (`begin` và `end`).

Chương 5: Code mẫu

```cpp
#include <algorithm>
#include <deque>
#include <iostream>
using namespace std;

int main() {
    deque<int> diem = {8, 2, 9};
    sort(diem.begin(), diem.end()); // {2, 8, 9}
    reverse(diem.begin(), diem.end()); // {9, 8, 2}

    auto it = find(diem.begin(), diem.end(), 8);
    if (it != diem.end()) cout << "Tim thay 8!";
    return 0;
}
```

BÀI KIỂM TRA CUỐI KHÓA

Thử thách cuối cùng

Giả sử bạn có hàng đợi `deque<int> q = {1, 2, 3, 4, 5};`.
Nếu thực hiện lệnh: `reverse(q.begin(), q.end());`
Sau đó gọi lệnh: `cout « q.front();`
Con số nào sẽ hiện ra?

BÀI KIỂM TRA CUỐI KHÓA

Thử thách cuối cùng

Giả sử bạn có hàng đợi `deque<int> q = {1, 2, 3, 4, 5};`.
Nếu thực hiện lệnh: `reverse(q.begin(), q.end());`
Sau đó gọi lệnh: `cout << q.front();`
Con số nào sẽ hiện ra?

Đáp án

Con số **5**. (Hàng bị đảo ngược thành {5, 4, 3, 2, 1}).

Chúc mừng bạn đã tốt nghiệp khóa học!