Khám phá std::map trong C++

Đọc Sách - Người đồng hành tri thức

2026

Lộ trình Khám phá "Chiếc túi Thần kỳ"Map

1. **Chương 1:** Chiếc hộp vạn năng là gì? (Key và Value)
2. **Chương 2:** Cách tạo ra chiếc hộp (Khai báo và Insertion)
3. **Chương 3:** Tìm đồ trong bóng tối (Truy xuất C++17/20)
4. **Chương 4:** Kiểm kê kho hàng (Structured Bindings)
5. **Chương 5:** Sức mạnh tiềm ẩn (Tại sao Map lại nhanh?)

Chương 1: Chiếc hộp vạn năng là gì?

Sự khác biệt giữa Vector và Map

- **Vector/Array:** Ngăn tủ đánh số $0, 1, 2...$ Phải nhớ số thứ tự.
- **Map:** Dán nhãn trực tiếp lên cửa tủ. Gọi tên nhãn, cửa tự mở.

Thành phần cốt lõi

- **Key (Khóa):** Nhãn dán duy nhất (ví dụ: "TEO").
- **Value (Giá trị):** Món đồ bên trong (ví dụ: điểm 10).

Ẩn dụ: Map giống như một cuốn từ điển. Từ vựng là Key, nghĩa của từ là Value.

Chương 2: Cách tạo ra chiếc hộp

Khai báo

```
map<Kiểu_Key, Kiểu_Value> tên_biến;
```

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

int main() {
    map<string, long long> danh_ba;

    // Cach 1: Gan truc tiep
    danh_ba["Nguyen Van A"] = 9876543210;

    // Cach 2: Dung insert (C++17)
    danh_ba.insert({"Le Van C", 555666777});

    // Cach 3: insert_or_assign (C++20)
    danh_ba.insert_or_assign("Nguyen Van A", 111222333);
    return 0;
}
```

Cơ chế tự sắp xếp của Map

Quản gia ngăn nắp

Mỗi khi bạn thêm một phần tử, `std::map` tự động sắp xếp các ngăn tủ theo **thứ tự từ điển (A-Z)** của các Key.

Câu hỏi tương tác

Giả sử tôi có đoạn code sau: `kho_banh["Banh Mi"] = 10;` `kho_banh["Banh Mi"] = 5;` Theo bạn, trong `kho_banh` có bao nhiêu ngăn tủ tên là "Banh Mi"?

Cơ chế tự sắp xếp của Map

Quản gia ngăn nắp

Mỗi khi bạn thêm một phần tử, `std::map` tự động sắp xếp các ngăn tủ theo **thứ tự từ điển (A-Z)** của các Key.

Câu hỏi tương tác

Giả sử tôi có đoạn code sau: `kho_banh["Banh Mi"] = 10;` `kho_banh["Banh Mi"] = 5;` Theo bạn, trong `kho_banh` có bao nhiêu ngăn tủ tên là "Banh Mi"? **Đáp án:** Chỉ có 1 ngăn duy nhất, giá trị mới (5) sẽ đè lên giá trị cũ (10).

Chương 3: Tìm đồ trong bóng tối

Kiểm tra tồn tại (C++20)

```
if (kho_banh.contains("Banh Mi")) {
    cout << "Co banh mi!";
}

```

Tìm và lấy dữ liệu (C++17)

```
if (auto it = kho_banh.find("Banh Mi"); it != kho_banh.end()) {
    cout << "So luong: " << it->second;
}

```

Cảnh báo nguy hiểm

Đừng dùng [] để kiểm tra sự tồn tại. Nó sẽ tự động tạo một ngăn tủ trống nếu Key chưa có!

Chương 4: Kiểm kê kho hàng

Structured Bindings (C++17)

Sử dụng "chiếc kéo thần kỳ"để tách đôi Key và Value ngay trong vòng lặp.

```
map<string, int> tui_do = {{"Kiem", 1}, {"Mau", 10}};

// Dung & de chay nhanh, const de bao ve du lieu
for (auto const& [ten, so_luong] : tui_do) {
    cout << "Ten: " << ten << " | SL: " << so_luong << endl;
}

```

Lợi ích

- Trực quan: Nhìn thấy `ten` và `so_luong` thay vì `first`, `second`.
- Ngăn nắp: Tự động in ra theo thứ tự A-Z của Key.

Chương 4.5: Iterator - Chiếc gậy chỉ đường

Duyệt thủ công bằng Iterator

```
map<string, int>::iterator it;
for (it = tui_do.begin(); it != tui_do.end(); ++it) {
    cout << it->first << " : " << it->second << endl;
}

```

Sức mạnh của auto

Thay vì viết `map<string, int>::iterator`, hãy dùng `auto`:

```
for (auto it = tui_do.begin(); it != tui_do.end(); ++it) { ... }

```

Chương 5: Sức mạnh tiềm ẩn

Tại sao Map lại nhanh?

Map không tìm kiếm tuần tự. Nó sử dụng cấu trúc **Cây tìm kiếm nhị phân (BST)**.

- **Độ phức tạp:** $O(\log n)$.
- **Ví dụ:** Với 1 triệu món đồ, Map chỉ cần tối đa khoảng 20 bước thử để tìm ra kết quả.

Khi nào nên dùng Map?

- Tra cứu dữ liệu (Từ điển, danh bạ).
- Đếm tần suất xuất hiện.
- Khi Key không phải là số nguyên (string, char, struct...).

Trận chiến cuối cùng: Quản lý Kho Vũ Khí

Thử thách cho bạn

Viết chương trình:

1. Tạo `map<string, int> vukhi`.
2. Thêm "Kiem"(50), "Cung"(40).
3. Dùng `auto &` để giảm sát thương tất cả đi 10.
4. Kiểm tra "Riu"bằng `contains`.

Trận chiến cuối cùng: Quản lý Kho Vũ Khí

Thử thách cho bạn

Viết chương trình:

1. Tạo `map<string, int> vukhi`.
2. Thêm "Kiem"(50), "Cung"(40).
3. Dùng `auto &` để giảm sát thương tất cả đi 10.
4. Kiểm tra "Riu"bằng `contains`.

```
for (auto & [ten, sat_thuong] : vukhi) {
    sat_thuong -= 10; // Su dung & de sua du lieu that
}

if (!vukhi.contains("Riu")) {
    vukhi["Riu"] = 60;
}

```

Tổng kết công thức "Vàng"

Ghi nhớ cụm từ: "Auto - Xem - Thật"

```
for (auto const & [k, v] : my_map)
```

- **Auto:** Tự nhận diện kiểu dữ liệu.
- **Xem (const):** Bảo vệ dữ liệu không bị sửa nhầm.
- **Thật (&):** Truy cập trực tiếp, không tốn công copy (Tăng tốc độ).