Chương I Vector

1. ****Khái niệm về vector****

* Trong C++, vector là một container mà thư viện tiêu chuẩn (STL - Standard Template Library) cung cấp. Nó cung cấp một cách linh hoạt để lưu trữ và quản lý dữ liệu theo dạng mảng động (dynamic array).

1. ****Khởi tạo vector****

* Có một số cách để khởi tạo vector trong C++:
* Khởi tạo vector rỗng: std::vector<int> myVector;
* Khởi tạo vector với kích thước cố định: std::vector<int> myVector(5); // vector có 5 phần tử
* Khởi tạo vector và gán giá trị cho các phần tử: std::vector<int> myVector = {1, 2, 3, 4, 5};

1. ****Truy cập phần tử của vector****

* Bạn có thể truy cập các phần tử của vector bằng cách sử dụng toán tử [] hoặc phương thức at().
* Ví dụ:
* int element = myVector[0]; // Truy cập phần tử đầu tiên
* int element = myVector.at(2); // Truy cập phần tử thứ 3

1. ****Thay đổi kích thước vector****

* Bạn có thể thay đổi kích thước của vector bằng cách sử dụng phương thức resize().
* Ví dụ:
* myVector.resize(10); // vector có 10 phần tử sau khi thay đổi kích thước

1. ****Thêm phần tử vào vector****

* Bạn có thể thêm phần tử vào vector bằng cách sử dụng phương thức push\_back().
* Ví dụ:
* myVector.push\_back(6); // Thêm phần tử 6 vào cuối vector
* Phương thức emplace\_back()
* Phương thức này được sử dụng để chèn một phần tử vào cuối vector bằng cách xây dựng phần tử trực tiếp trong vector. Điều này tiết kiệm một bước sao chép so với phương thức push\_back().
* Ví dụ:
* std::vector<std::string> myVector;
* myVector.emplace\_back("Hello"); // Thêm một chuỗi "Hello" vào cuối vector
* Phương thức insert():
* Phương thức này cho phép chèn một phần tử vào vị trí bất kỳ trong vector bằng cách xác định vị trí và giá trị của phần tử cần chèn.
* Ví dụ:
* std::vector<int> myVector = {1, 2, 3, 4, 5};
* myVector.insert(myVector.begin() + 2, 10); // Chèn phần tử 10 vào vị trí thứ 2 trong vector
* Phương thức assign():
* Phương thức này được sử dụng để gán một loạt các phần tử cho vector từ một phạm vi hoặc danh sách khác.
* Ví dụ:
* std::vector<int> myVector;
* myVector.assign({1, 2, 3, 4, 5}); // Gán các phần tử từ danh sách cho vector
* Phương thức resize():
* Phương thức này được sử dụng để thay đổi kích thước của vector và cũng có thể được sử dụng để thêm các phần tử mới vào vector.
* Ví dụ:
* std::vector<int> myVector = {1, 2, 3};
* myVector.resize(5, 0); // Thay đổi kích thước vector thành 5 và các phần tử mới có giá trị 0

1. ****Xoá phần tử khỏi vector****

* Bạn có thể xoá phần tử khỏi vector bằng cách sử dụng phương thức erase().
* Ví dụ:
* myVector.erase(myVector.begin() + 2); // Xoá phần tử thứ 3

1. ****Lấy kích thước vector****

* Bạn có thể lấy kích thước của vector bằng cách sử dụng phương thức size().
* Ví dụ:
* int size = myVector.size(); // Lấy kích thước của vector

Chương II List

1. Khái niệm về list

* Trong C++, list là một container mà thư viện tiêu chuẩn (STL - Standard Template Library) cung cấp. List là một dạng danh sách liên kết hai chiều, cho phép thêm, xoá và truy cập các phần tử một cách hiệu quả
* Để sử dụng list trong C++, bạn cần bao gồm thư viện <list> bằng cách khai báo "#include <list>" ở đầu chương trình.

1. Khởi tạo list

* Có một số cách để khởi tạo list trong C++
* Khởi tạo list rỗng: std::list<int> myList;
* Khởi tạo list với các giá trị ban đầu: std::list<int> myList = {1, 2, 3, 4, 5};

1. Truy cập phần tử của list

* Bạn có thể truy cập các phần tử của list bằng cách sử dụng iterator.
* Ví dụ:
* std::list<int>::iterator it = myList.begin(); // Iterator trỏ đến phần tử đầu tiên
* int element = \*it; // Lấy giá trị của phần tử đầu tiên

1. Thêm phần tử vào list

* Phương thức push\_back(): Thêm một phần tử vào cuối list.
* Phương thức push\_front(): Thêm một phần tử vào đầu list.
* Phương thức insert(): Chèn một phần tử vào vị trí bất kỳ trong list.
* Ví dụ:
* std::list<int> myList;
* myList.push\_back(5); // Thêm phần tử 5 vào cuối list
* myList.push\_front(3); // Thêm phần tử 3 vào đầu list
* myList.insert(std::next(myList.begin()), 4); // Chèn phần tử 4 vào vị trí thứ 2 trong list

1. Xoá phần tử khỏi list: Có một số cách để xoá phần tử khỏi list:

* Phương thức pop\_back(): Xoá phần tử cuối cùng trong list.
* Phương thức pop\_front(): Xoá phần tử đầu tiên trong list.
* Phương thức erase(): Xoá một hoặc một khoảng phần tử trong list.
* Ví dụ:
* std::list<int> myList = {1, 2, 3, 4, 5};
* myList.pop\_back(); // Xoá phần tử cuối cùng trong list
* myList.pop\_front(); // Xoá phần tử đầu tiên trong list
* myList.erase(std::next(myList.begin()), std::prev(myList.end())); // Xoá các phần tử từ vị trí thứ 2 đến