# C++ vector 容器

C++ 中的 vector 是一种序列容器，它允许你在运行时动态地插入和删除元素。

vector 是基于数组的数据结构，但它可以自动管理内存，这意味着你不需要手动分配和释放内存。

与 C++ 数组相比，vector 具有更多的灵活性和功能，使其成为 C++ 中常用的数据结构之一。

vector 是 C++ 标准模板库（STL）的一部分，提供了灵活的接口和高效的操作。

**基本特性:**

- **动态大小**：`vector` 的大小可以根据需要自动增长和缩小。
- **连续存储**：`vector` 中的元素在内存中是连续存储的，这使得访问元素非常快速。
- **可迭代**：`vector` 可以被迭代，你可以使用循环（如 `for` 循环）来访问它的元素。
- **元素类型**：`vector` 可以存储任何类型的元素，包括内置类型、对象、指针等。

**使用场景:**

- 当你需要一个可以动态增长和缩小的数组时。
- 当你需要频繁地在序列的末尾添加或移除元素时。
- 当你需要一个可以高效随机访问元素的容器时。

要使用 vector，首先需要包含 **`<vector>`** 头文件：

```
#include <vector>
```

## 创建 Vector

创建一个 vector 可以像创建其他变量一样简单：

```
std::vector<int> myVector; // 创建一个存储整数的空 vector
```

这将创建一个空的整数向量,也可以在创建时指定初始大小和初始值:

```
std::vector<int> myVector(5); // 创建一个包含 5 个整数的 vector，每个值都为默认值（0）
std::vector<int> myVector(5, 10); // 创建一个包含 5 个整数的 vector，每个值都为 10
```

或:

```
std::vector<int> vec; // 默认初始化一个空的 vector
std::vector<int> vec2 = {1, 2, 3, 4}; // 初始化一个包含元素的 vector
```

## 添加元素

可以使用 push_back 方法向 vector 中添加元素:

```
myVector.push_back(7); // 将整数 7 添加到 vector 的末尾
```

## 访问元素

可以使用下标操作符 [] 或 at() 方法访问 vector 中的元素:

```
int x = myVector[0]; // 获取第一个元素
int y = myVector.at(1); // 获取第二个元素
```

## 获取大小

可以使用 size() 方法获取 vector 中元素的数量:

```
int size = myVector.size(); // 获取 vector 中的元素数量
```

## 迭代访问

可以使用迭代器遍历 vector 中的元素：

```
for (auto it = myVector.begin(); it != myVector.end(); ++it) {
    std::cout << *it << " ";
}
```

或者使用范围循环：

```
for (int element : myVector) {
    std::cout << element << " ";
}
```

## 删除元素

可以使用 erase() 方法删除 vector 中的元素：

```
myVector.erase(myVector.begin() + 2); // 删除第三个元素
```

## 清空 Vector

可以使用 clear() 方法清空 vector 中的所有元素：

```
myVector.clear(); // 清空 vector
```

# 实例

以下是一个完整的使用实例，包括创建 vector、添加元素、访问元素以及输出结果的代码：

**实例**

```cpp
#include <iostream>
#include <vector>

int main() {
    // 创建一个空的整数向量
    std::vector<int> myVector;

    // 添加元素到向量中
    myVector.push_back(3);
    myVector.push_back(7);
    myVector.push_back(11);
    myVector.push_back(5);

    // 访问向量中的元素并输出
    std::cout << "Elements in the vector: ";
    for (int element : myVector) {
        std::cout << element << " ";
    }
    std::cout << std::endl;

    // 访问向量中的第一个元素并输出
    std::cout << "First element: " << myVector[0] << std::endl;

    // 访问向量中的第二个元素并输出
    std::cout << "Second element: " << myVector.at(1) << std::endl;

    // 获取向量的大小并输出
    std::cout << "Size of the vector: " << myVector.size() << std::endl;

    // 删除向量中的第三个元素
    myVector.erase(myVector.begin() + 2);

    // 输出删除元素后的向量
    std::cout << "Elements in the vector after erasing: ";
    for (int element : myVector) {
        std::cout << element << " ";
    }
    std::cout << std::endl;
```

```
    // 清空向量并输出
    myVector.clear();
    std::cout << "Size of the vector after clearing: " << myVector.size() << std::endl;

    return 0;
}
```

以上代码创建了一个整数向量，向其中添加了几个元素，然后输出了向量的内容、元素的访问、向量的大小等信息，接着删除了向量中的第三个元素，并输出删除元素后的向量。最后清空了向量，并输出清空后的向量大小。

输出结果为：

```
Elements in the vector: 3 7 11 5
First element: 3
Second element: 7
Size of the vector: 4
Elements in the vector after erasing: 3 7 5
Size of the vector after clearing: 0
```