C++中vector的介绍

1 简介

在C++中，`vector` 是标准模板库（STL）中最常用的顺序容器之一。它可以理解为一个动态数组，能够根据需要自动调整大小，并且提供了快速的随机访问和高效的尾部插入操作。

2 vector的特点

- **动态大小**：`vector` 可以自动扩展或收缩，不需要提前指定大小。
- **随机访问**：支持常数时间的随机访问，类似于普通数组。
- **自动管理内存**：`vector` 自动管理其内存分配，用户不需要手动分配和释放内存。
- **高效的尾部操作**：在末尾插入、删除元素的时间复杂度是常数 $O(1)$，但在中间或开头插入、删除元素的操作时间复杂度为线性 $O(n)$。

3 常见操作

`vector` 提供了丰富的操作函数，常见操作包括：

3.1 初始化与定义

- 默认构造函数：`vector<int> v;`
- 初始化指定大小：`vector<int> v(5);` 初始化大小为5，所有元素默认初始化为0。
- 初始化大小和默认值：`vector<int> v(5, 10);` 初始化大小为5，每个元素的值都是10。
- 列表初始化：`vector<int> v = {1, 2, 3, 4, 5};`

3.2 添加元素

- `push_back`：在末尾添加元素。
- `insert`：在指定位置插入元素。

3.3 删除元素

- `pop_back`：删除最后一个元素。
- `erase`：删除指定位置的元素或范围。

3.4 访问元素

- 使用索引：v[i]
- 使用 at 函数：v.at(i)，这种方式会进行边界检查。
- 获取第一个和最后一个元素：front() 和 back()。

3.5 遍历元素

- 使用迭代器遍历：begin() 和 end()。
- 使用范围 for 循环。

3.6 容量管理

- size()：当前元素个数。
- capacity()：vector 容器的当前容量。
- resize()：调整容器大小。
- reserve()：调整容量，预留空间。

4 示例代码

下面展示一些使用 vector 的代码示例。

4.1 基本操作示例

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

int main() {
    vector<int> v;

    // 添加元素
    v.push_back(10);
    v.push_back(20);
    v.push_back(30);

    // 输出元素
    cout << "Vector elements: ";
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";
    }
    cout << endl;
```

```
    // 使用 at 函数访问
    cout << "Element at index 1: " << v.at(1) << endl;

    // 获取第一个和最后一个元素
    cout << "First element: " << v.front() << endl;
    cout << "Last element: " << v.back() << endl;

    // 删除最后一个元素
    v.pop_back();
    cout << "After pop_back, size: " << v.size() << endl;

    return 0;
}
```

4.2 使用迭代器遍历 vector

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

int main() {
    vector<int> v = {1, 2, 3, 4, 5};

    // 使用迭代器遍历
    cout << "Using iterator to print elements: ";
    for (vector<int>::iterator it = v.begin(); it != v.end(); ++it) {
        cout << *it << " ";
    }
    cout << endl;

    return 0;
}
```

4.3 插入和删除元素

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

int main() {
    vector<int> v = {1, 2, 3, 4, 5};

    // 在指定位置插入元素
```

```
    v.insert(v.begin() + 2, 10);  // 在索引 2 处插入 10

    // 删除指定位置的元素
    v.erase(v.begin() + 4);  // 删除索引 4 处的元素

    // 输出修改后的 vector
    cout << "Modified vector: ";
    for (int num : v) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

5 总结

vector 是一个功能强大且灵活的动态数组，它提供了方便的内存管理和常见的操作函数。在使用 vector 时，不仅可以享受类似数组的高效随机访问，还可以方便地动态调整其大小，非常适合处理需要频繁插入、删除和遍历的场景。