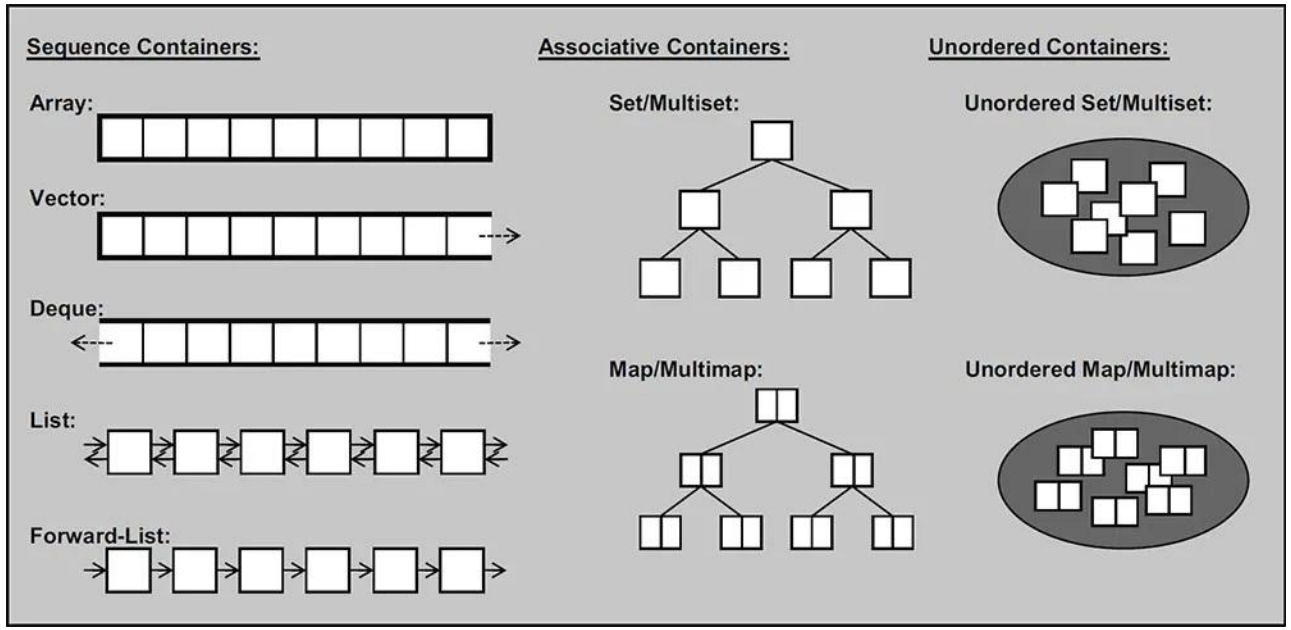
STL模板库及其应用

汪船长

初航我带你，远航靠自己

STL容器

STL（Standard Template Library，标准模板库）容器是 C++ 标准库提供的一系列类模板，其本质是用于存储和管理数据的数据结构。



vector

特点：vector 是一个动态数组，能够在运行时改变大小。它支持随机访问，可通过下标快速访问元素。

常用操作：

push_back()：在容器尾部添加元素。

pop_back()：移除容器尾部的元素。

size()：返回容器中元素的数量。

[]：通过下标访问元素。

vector

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

int main() {
    vector<int> vec;
    vec.push_back(1);
    vec.push_back(2);
    vec.push_back(3);
    for (int i = 0; i < vec.size(); ++i) {
        cout << vec[i] << " ";
    }
    cout << endl;
    return 0;
}

```

queue

特点：queue 是一个先进先出（FIFO）的数据结构，只允许在队尾插入元素，在队头删除元素。

常用操作：

push()：在队尾插入元素。

pop()：移除队头的元素。

front()：返回队头的元素。

back()：返回队尾的元素。

size()：返回队列中元素的数量。

queue

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

int main() {
    queue<int> q;
    q.push(1);
    q.push(2);
    q.push(3);
    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
    cout << endl;
    return 0;
}
```

stack

特点：stack 是一个后进先出（LIFO）的数据结构，只允许在栈顶插入和删除元素。

常用操作：

push()：在栈顶插入元素。

pop()：移除栈顶的元素。

top()：返回栈顶的元素。

size()：返回栈中元素的数量。

stack

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

int main() {
    stack<int> s;
    s.push(1);
    s.push(2);
    s.push(3);
    while (!s.empty()) {
        cout << s.top() << " ";
        s.pop();
    }
    cout << endl;
    return 0;
}
```

deque

特点：deque 是双端队列，支持在队列的两端高效地插入和删除元素，同时也支持随机访问。

常用操作：

push_front()：在队列头部插入元素。

push_back()：在队列尾部插入元素。

pop_front()：移除队列头部的元素。

pop_back()：移除队列尾部的元素。

[]：通过下标访问元素。

deque

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

int main() {
    deque<int> dq;
    dq.push_back(1);
    dq.push_front(2);
    dq.push_back(3);
    for (int i = 0; i < dq.size(); ++i) {
        cout << dq[i] << " ";
    }
    cout << endl;
    return 0;
}
```

priority_queue

特点：priority_queue 是优先队列，元素按照优先级排列，优先级高的元素先出队。

默认情况下，最大元素具有最高优先级。

常用操作：

push()：插入元素。

pop()：移除优先级最高的元素。

top()：返回优先级最高的元素。

size()：返回队列中元素的数量。

priority_queue

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

int main() {
    priority_queue<int> pq;
    pq.push(3);
    pq.push(1);
    pq.push(2);
    while (!pq.empty()) {
        cout << pq.top() << " ";
        pq.pop();
    }
    cout << endl;
    return 0;
}
```

使用优先队列实现小根堆

默认的比较规则是 <（小于号），但是我们也可以在定义 priority_queue 的时候将规则进行修改。比如下面的代码段就定义了一个大根堆：

```
priority_queue<int, vector<int>, greater<int> > que;
```

其中，priority_queue 后的尖括号中：

1. int 表示数据类型；
2. vector<int> 表示数据的存储方式，在这里是使用 vector 存储；
3. greater<int> 表示比较规则，这个 greater<int> 对应的比较规则就是 >（大于号），即 “我比你大，我把你顶上去”

需要注意的是，如果 priority_queue 存储的是别的类型的数据，则对应的数据类型都得进行相应的修改，如下下面的代

```
priority_queue<double, vector<double>, greater<double> > que;
```

使用优先队列实现小根堆

```cpp
#include <bits/stdc++.h>
using namespace std;
priority_queue<int, vector<int>, greater<int> > que;
int main() {
    for (int i = 3; i <= 6; i ++)
        que.push(i);
    que.push(1);
    que.push(8);
    cout << "size = " << que.size() << endl;
    while (!que.empty()) {
        cout << que.top() << ",";
        que.pop();
    }
    return 0;
}
```

优先队列+结构体

对于结构体类型的变量来说，默认没有 <（小于号），这种情况下直接使用该结构体类型的 priority_queue 显然是不行的（会报错）。

所以可以考虑为新定义的结构体类型定义一个 < 的功能，这种操作被称作 重载运算符。

```
struct Node {
    int x, y;

    bool operator < (const Node b) const {
        return x < b.x || x == b.x && y < b.y;
    }
};
```

优先队列+结构体

```cpp
#include <bits/stdc++.h>
using namespace std;
struct Node {
    int x, y;

    bool operator < (const Node b) const {
        return x < b.x || x == b.x && y < b.y;
    }
};
priority_queue<Node> que;

int main() {
    que.push({3, 5});
    que.push({2, 4});
    que.push({1, 3});
    que.push({4, 2});
    que.push({3, 3});
    while (!que.empty()) {
        Node u = que.top();
        que.pop();
        cout << "(" << u.x << " , " << u.y << ")" << endl;
    }
    return 0;
}
```

自定义优先队列的比较规则

有的时候，有的数据类型可能已经封装好了 < 运算符，或者其 < 运算符还有别的用处，这种情况下我们不能再为其重载 < 运算符，那么这个时候怎么办呢？

回顾一下，在使用 sort 函数的时候，我们定义过比较函数（一般取名为 cmp，国际惯例）。

然后在 sort 的时候将 cmp 函数作为 sort 函数的第三个参数。

在 priority_queue 中也可以使用类型的功能，只不过 priority_queue 使用的是 **比较结构体**。

我们可以定义一个名为 Cmp 的结构体并重载其 () 运算符，然后将其作为 priority_queue 定义时尖括号中的第三个参数。

```
struct Cmp {
    bool operator () (Node &a, Node &b) {
        return a.x < b.x || a.x == b.x && a.y < b.y;
    }
};
```

自定义优先队列的比较规则

```cpp
#include <bits/stdc++.h>
using namespace std;
struct Node {
    int x, y;
};
struct Cmp {
    bool operator () (Node &a, Node &b) {
        return a.x < b.x || a.x == b.x && a.y < b.y;
    }
};
priority_queue<Node, vector<Node>, Cmp> que;
int main() {
    que.push({3, 5});
    que.push({2, 4});
    que.push({1, 3});
    que.push({4, 2});
    que.push({3, 3});
    while (!que.empty()) {
        Node u = que.top();
        que.pop();
        cout << "(" << u.x << " , " << u.y << ")" << endl;
    }
    return 0;
}
```

map

特点：map 是关联容器，存储键值对，键是唯一的，且元素按照键的顺序排列。

常用操作：

insert()：插入键值对。

[]：通过键访问或插入值。

find()：查找指定键的元素。

erase()：移除指定键的元素。

map

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

int main() {
    map<string, int> m;
    m["apple"] = 1;
    m["banana"] = 2;
    m["cherry"] = 3;
    for (auto it = m.begin(); it != m.end(); ++it) {
        cout << it->first << ": " << it->second << endl;
    }
    return 0;
}
```

set

特点：set 是关联容器，存储唯一的元素，元素按照升序排列。

常用操作：

insert()：插入元素。

find()：查找指定元素。

erase()：移除指定元素。

size()：返回集合中元素的数量。

set

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

int main() {
    set<int> s;
    s.insert(3);
    s.insert(1);
    s.insert(2);
    for (auto it = s.begin(); it != s.end(); ++it) {
        cout << *it << " ";
    }
    cout << endl;
    return 0;
}
```

multiset

特点：multiset 与 set 类似，但允许存储重复的元素，元素同样按照升序排列。

常用操作：

insert()：插入元素。

find()：查找指定元素。

erase()：移除指定元素。

size()：返回集合中元素的数量。

multiset

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

int main() {
    multiset<int> ms;
    ms.insert(3);
    ms.insert(1);
    ms.insert(2);
    ms.insert(2);
    for (auto it = ms.begin(); it != ms.end(); ++it) {
        cout << *it << " ";
    }
    cout << endl;
    return 0;
}
```

STL算法

STL 算法是 C++ 标准模板库中一组通用的、独立于具体数据类型和容器的函数模板，用于对容器中的元素进行诸如查找、排序、替换、遍历等各种操作。

常见的一些STL库函数

1. sort：sort(first, last, comp?) 对 [first, last) 排序，comp 可选，默认升序。
2. lower_bound：在 [first, last) 找首个不小于 val 的元素，返回迭代器。
3. upper_bound：在 [first, last) 找首个大于 val 的元素，返回迭代器。
4. find：在 [first, last) 找等于 val 的元素，返回迭代器，未找到则为 last。
5. nth_element：重排 [first, last) 使第 n 小元素就位，左边小右边大。

常见的一些STL库函数

6. unique：移除 [first, last) 相邻重复元素，返回新逻辑末尾迭代器。
7. reverse：反转 [first, last) 元素顺序。
8. fill：将 [first, last) 元素赋值为 val。
9. copy：把 [first, last) 元素复制到以 result 开始处。
10. shuffle：随机打乱 [first, last) 元素顺序。
11. next_permutation：生成 [first, last) 下一个字典序排列，成功返回 true。

十七、STL模板库及其应用

1. XYOJ-7303.模板库应用1-模板题1
2. XYOJ-7305.模板库应用1-模板题3
3. XYOJ-7308.模板库应用1-巩固题1
4. XYOJ-7317.模板库应用1-趣味题1
5. XYOJ-7300.模板库应用2-模板题1
6. XYOJ-7306.模板库应用2-巩固题1
7. XYOJ-7307.模板库应用2-巩固题2
8. XYOJ-15383.Guess the Animal
9. XYOJ-7669.日记账单