C++ STL之set

在C++中，`set`是标准模板库（STL）中的一种关联容器，基于红黑树实现，具有以下核心特性：

核心特性

- **唯一性**：存储的元素必须唯一，自动去除重复项。
- **有序性**：元素默认按升序排列（可通过自定义比较函数调整）。
- **高效操作**：插入、删除、查找操作时间复杂度为$O(logn)$。

典型应用

适用于需要高效检索且元素唯一的场景，例如：

- 快速查找数据
- 维护动态排序列表
- 去除重复数据

需注意，`set`仅存储键值对中的键（value作为唯一标识），不修改元素值本身。

一. 常用方法

1. 定义set  007794DA  007794DA  007794DA

与vector类似，我们使用set之前需要先定义，set一共有三种定义方式，如下代码所示：

```cpp
// 创建一个空的、用于存储int的set
 set<int> s1;

 // 创建并初始化
 set<string> s2 = {"apple", "banana", "orange"};

 // 创建一个降序排列的set（使用greater仿函数）
 set<double, greater<double>> s3;
```

重点是**第一个和第三个**

2.插入元素   

增加元素是通过.insert(key)方法，当插入的数据已经存在时，则忽略本次操作。

```cpp
set<int> s;

s.insert(3);
s.insert(1);
s.insert(4);
s.insert(1); //这次插入是无效的，因为1已经存在

// 此时set中的元素为：1, 3, 4
```

3.删除元素  

删除元素也有两种方式，如下代码所示：

```
s.erase(3); // 删除值为3的元素

// 也可以传递迭代器来删除指定位置的元素
auto it = s.find(3);
if(it != s.end()){
    s.erase(it);
}
```

以上两种方式都可以删除，一个是直接删除，一个是查询后再删除。
如果查询后再删除，一定要注意先判断是否等于s.end()，再删除。

通常第二种用的频率较低，**学会第一种即可**

4.查找元素 007794DA

使用 `s.find(key)`。
如果找到，返回指向该元素的迭代器；
如果没找到，返回 `s.end()`。
注意.end()表示set集合中的最后一个元素的下一个位置，相当于哨兵，是一个虚拟的位置。
因此要用 !=s.end() 而不是 <=s.end() 。

```
auto pos = s.find(5);
if (pos != s.end()){
   cout << "找到了" << *pos << endl;
} else {
   cout << "未找到" << endl;
}
```

使用频率低，因为如果单纯判断元素是否存在，我们一般不用find方法，而且返回的结果是该元素的迭代器，使用起来也比较麻烦。

5.判断元素是否存在  007794DA  007794DA  007794DA

使用 `s.count(key)`。由于set元素唯一，返回值只能是 **1（存在）或 0（不存在）**。

```
if(s.count(5)){
    cout << "元素存在。" << endl;
}
```

6.其他常用方法

`.empty()`：判断set是否为空。

`.size()`：返回set中元素的个数。

二.竞赛中常用场景

1.快速去重并排序

```
// 输入：5, 3, 5, 2, 3, 1

set<int> s;
int n;
while(cin >> n){
    s.insert(n);
}

// 输出s即可得到：1, 2, 3, 5
```

2.判断元素是否存在

在一些需要频繁检查某个值是否出现在一个集合中的题目，使用`set`的`find()`或`count()`比在数组中遍历要高效得多。

3.维护有序数据

有些题目需要动态维护一个有序序列，并随时进行插入、删除和查询。使用set可以避免手动维护排序的麻烦。

三.注意事项

set中的元素只读

set中的元素是只读的：一旦元素被放入set，就不能直接修改它。

因为修改可能会破坏红黑树的结构。
如果需要修改，通常的做法是先删除旧元素，再插入新元素。