set容器

概述

set 翻译为集合，是一个内部自动有序且不含重复元素的容器。

当出现需要去掉重复元素的情况，而且有可能因这些元素比较大或者类型不是 int 型而不能直接开散列表，在这种情况下就可以用 set 来保留元素本身而不考虑它的个数。

当然，上面说的情况也可以通过再开一个数组进行下标和元素的对应来解决，但是 set 提供了更为直观的接口，并且加入 set 之后可以实现自动排序，默认是升序排列。

基本用法

1、头文件

```
#include<set>
```

2、定义

定义单个set

set<typename> name;

//这里的typename可以是任何基本类型，例如 int、double、char、结构体等，也可以是STL标准容器，例如vector、set、queue等。

```
set<int> set1;
set<double> set2;
set<char> set3;
//如果typename 是一个STL容器，那么定义时要记得在 >>之间要加空格
set<vector<int> > set4;
```

set数组

```
set<int> vi[100];
```

3、元素访问

set只能通过迭代器访问

```
set<int>::iterator it;
set<double>::iterator it;
//这样就得到了迭代器 it ,并且可以通过 *it 来访问 set 里元素。

set<int> st;
for (int i = 1; i <= 5; i++) {
```

```
        st.insert(i);
    }
    for (set<int>::iterator it = st.begin(); it != st.end(); it++) {
        printf("%d ", *it);
    }
```

4、常用函数

insert(x)：可将 x 插入 set 容器中，并自动递增排序和去重

find(value)：返回 set 中对应值为 value 的迭代器，时间复杂度为 O（logN）

```
    set<int> st;
    for (int i = 5; i >= 1; i--) {
        st.insert(i);
    }
    //在 set 中查找2，返回其迭代器。
    set<int>::iterator it = st.find(2);
    printf("%d ", *it);
```

erase()

4-1. 删除单个元素。

删除单个元素有两种方法：

（1）st.erase(it) ，即删除迭代器为 it 处的元素，时间复杂度为 O（1）。可以结合 find() 函数来使用。示例如下：

```
    #include<bits/stdc++.h>

    using namespace std;

    int main() {
        set<int> st;
        for (int i = 5; i >= 1; i--) {
            st.insert(i * 100);
        }
        //利用 find() 函数找到200，然后用 erase 删除它。
       st.erase(st.find(200));
        for (set<int>::iterator it = st.begin(); it != st.end(); it++)
            printf("%d \n", *it);
        }
        return 0;
    }
```

（2）st.erase(value) ，value 为所需要删除元素的值。时间复杂度为 O（logN），N 为 set 内的元素个数。示例如下：

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    set<int> st;
    for (int i = 5; i >= 1; i--) {
        st.insert(i * 100);
    }
    //用 erase 删除200。
   st.erase(200);
    for (set<int>::iterator it = st.begin(); it != st.end(); it++)
        printf("%d \n", *it);
    }
    return 0;
}
```

4-2. 删除一个区间内的所有元素。erase(first,last) 即删除 [first,last) 内的所有元素。其中 first 为所需要删除区间的起始迭代器，而 last 则为所需要删除区间的末尾迭代器的下一个地址。

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    set<int> st;
    for (int i = 5; i >= 1; i--) {
        st.insert(i * 100);
    }
    st.erase(st.find(300), st.end());
    for (set<int>::iterator it = st.begin(); it != st.end(); it++)
        printf("%d \n", *it);
    }
    return 0;
}
```

size：用来获得 set 中元素的个数，时间复杂度为 O（1）。size()返回的是 unsigned 类型，不过一般来说用 %d 不会出很大问题，这一点对所有 STL 容器都是一样的。

empty()：判断set是否为空

clear()：用来清空set中的所有元素。

size()：返回set中的元素个数

begin()：获取第一个元素的迭代器

end()：获取最后一个元素下一个位置的迭代器

lower_bound(x)：返回一个迭代器，指向第一个键值不小于x的元素

up_bound(x)：返回一个迭代器，指向第一个键值大于x的元素

{

{

{