关联容器

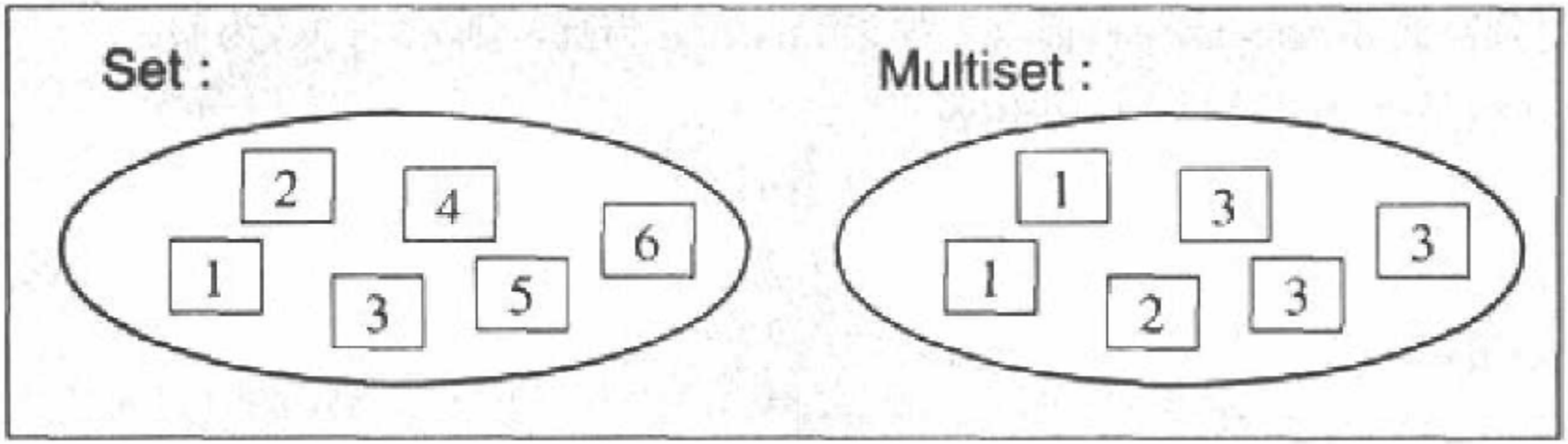
瞿绍军
湖南师范大学

关联容器

- 关联容器的元素可自动按照某种标准进行排序，默认的排序标准是“<”
- 采用二叉搜索树实现
- 对于每一个元素，父节点的键值比左儿子节点的键值大，比右儿子节点的键值小。

set 和 multiset

set 和 multiset会根据特定的排序准则，自动将元素排序，两者的不同之处在于multiset可以允许元素重复而set不允许元素重复。



set和multiset

头文件: #include <set>

定义：set <data_type> set_name;

如：set <int> s;//默认由小到大排序

如果想按照自己的方式排序，可以重载小于号。

```
struct new_type{
        int x, y;
        bool operator < (const new_type &a)const{
                if(x != a.x) return x < a.x;
                return y < a.y;
        }
}
set <new_type> s;
```

set和multiset

声明：

```
set<elmType> ct;
set<elmType,sortOp> ct
set<elmType> ct(otherCt);
set<elmType,sortOp> ct(otherCt);
set<elmType> ct(beg,end);
set<elmType,sortOp> ct(beg,end);
```

set和multiset

操作：

s.insert(elem) -- 安插一个elem副本，返回新元素位置。

s.erase(elem) -- 移除与elem元素相等的所有元素，返回被移除的元素个数。

s.erase(pos) -- 移除迭代器pos所指位置上的元素，无返回值。

s.clear() -- 移除全部元素，将整个容器清空。

迭代器举例：

```
multiset <int> :: iterator pos;
for(pos = s.begin(); pos != s.end(); pos++)
    ... ...
```

set和multiset

操作：

s.size() -- 返回容器大小。

s.empty() -- 返回容器是否为空。

s.count(elem) -- 返回元素值为elem的元素的个数。

s.lower_bound(elem) -- 返回elem的第一个可安插的位置。

也就是“元素值>= elem的第一个元素位置”。

s.upper_bound(elem) -- 返回elem的最后一个可安插的位置。

也就是“元素值 > elem的第一个元素的位置”。

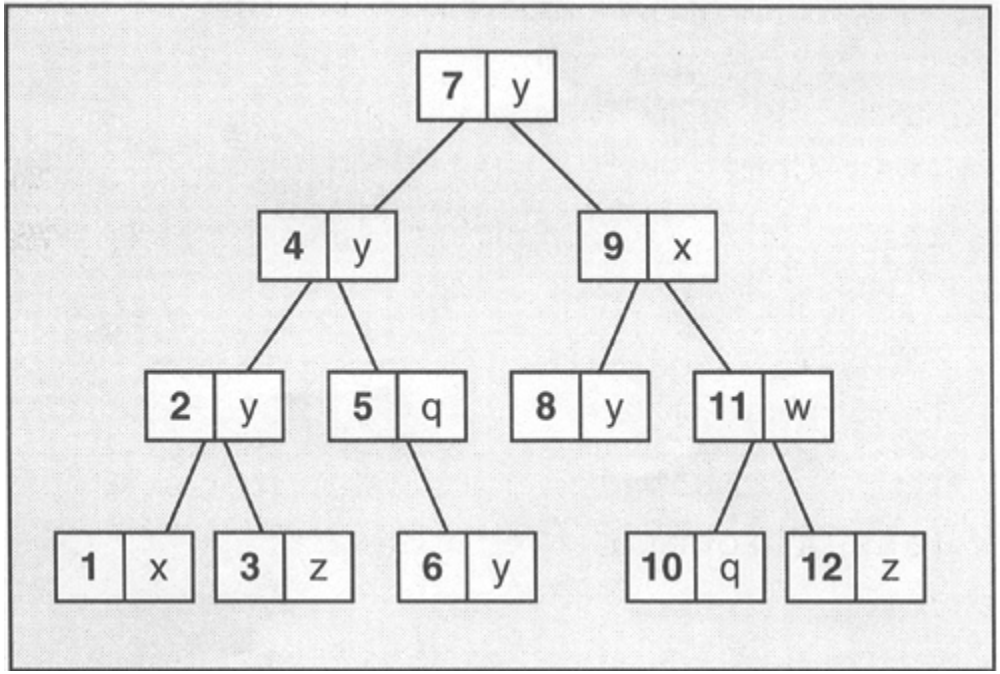
以上位置均为一个迭代器。

s.begin() -- 返回一个双向迭代器，指向第一个元素。

s.end() -- 返回一个双向迭代器，指向最后一个元素的下一个位置

map和multimap

所有元素都会根据元素的键值自动排序，map的所有元素都是pair，pair的第一个元素被视为键值，第二个元素为实值。map不允许两个元素有相同的键值，但multimap可以。



map和multimap 简单应用

头文件: #include <map>

定义：map <data_type1, data_type2> map_name;

如：map <string, int> m;//默认按string由小到大排序

map<key,elmType> ct;
map<key,elmType,sortOp> ct
map<key,elmType> ct(otherCt);
map<key,elmType,sortOp> ct(otherCt);
map<key,elmType> ct(beg,end);
map<key,elmType,sortOp> ct(beg,end);

map和multimap

操作：

m.size() 返回容器大小

m.empty() 返回容器是否为空

m.count(key) 返回键值等于key的元素的个数

m.lower_bound(key) 返回键值等于key的元素的第一个可安插 的位置

m.upper_bound(key) 返回键值等于key的元素的最后一个可安插的位置

map和multimap

操作：

m.begin() 返回一个双向迭代器，指向第一个元素。

m.end() 返回一个双向迭代器，指向最后一个元素的下一个位置。

m.clear() 讲整个容器清空。

m.erase(elem) 移除键值为elem的所有元素，返回个数，对于map来说非0即1。

m.erase(pos) 移除迭代器pos所指位置上的元素。

直接元素存取：

m[key] = value；

查找的时候如果没有键值为key的元素，则安插一个键值为key的新元素，实值为默认(一般0)。

map和multimap

操作：

m.insert(elem) 插入一个元素elem

a)运用value_type插入

```
map<string, float> m;
m.insert(map<string, float>:: value_type
("Robin", 22.3));
```

b) 运用pair<>

```
m.insert(pair<string, float>("Robin", 22.3));
```

c) 运用make_pair()

```
m.insert(make_pair("Robin", 22.3));
```

容器、相关头文件、每种容器支持的迭代器

序列容器	相关头文件	支持的迭代器类型
向量	<vector>	随机访问迭代器
双端队列	<deque>	随机访问迭代器
表	<list>	双向迭代器
关联容器	相关头文件	支持的迭代器类型
集合	<set>	双向迭代器
多重集合	<set>	双向迭代器
映射	<map>	双向迭代器
多重映射	<map>	双向迭代器
适配器	相关头文件	支持的迭代器类型
堆栈	<stack>	不支持任何迭代器
队列	<queue>	不支持任何迭代器
优先级队列	<queue>	不支持任何迭代器