程序设计实习

郭炜　微博 http://weibo.com/guoweiofpku
http://blog.sina.com.cn/u/3266490431
刘家瑛 微博 http://weibo.com/pkuliujiaying

标准模板库STL

map和multimap

multiset非常适合需要不断的添加数据和不断的查询的情况。插入和查询的时间复杂度都是LOG(N)

预备知识： pair 模板

```
template<class _T1, class _T2>
  struct pair
  {
    typedef _T1 first_type;
    typedef _T2 second_type;
    _T1 first;
    _T2 second;
    pair(): first(), second() { }
    pair(const _T1& __a, const _T2& __b)
    : first(__a), second(__b) { }
    template<class _U1, class _U2>
    pair(const pair<_U1, _U2>& __p)
            : first(__p.first), second(__p.second) { }
  };
```

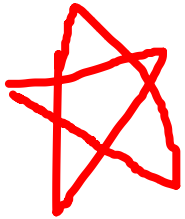
map/multimap里放着的都是pair模版类的对象，且按first从小到大排序

第三个构造函数用法示例：

```
pair<int,int>
p(pair<double,double>(5.5,4.6));
// p.first = 5, p.second = 4
```

multimap

```
template<class Key, class T, class Pred = less<Key>,
         class A = allocator<T> >
class multimap {
    ....
    typedef pair<const Key, T> value_type;
    .......
};   //Key 代表关键字的类型
```



- multimap中的元素由 <关键字,值>组成，每个元素是一个pair对象，关键字就是first成员变量,其类型是Key
- multimap 中允许多个元素的关键字相同。元素按照first成员变量从小到大排列，缺省情况下用 less<Key> 定义关键字的“小于”关系。

multimap示例

```
#include <iostream>
#include <map>
using namespace std;
int main()    {
        typedef multimap<int,double,less<int> > mmid;
        mmid pairs;
        cout << "1) " << pairs.count(15) << endl;
        pairs.insert(mmid::value_type(15,2.7));//typedef pair<const Key, T> value_type;
        pairs.insert(mmid::value_type(15,99.3));
        cout << “2) ” << pairs.count(15) << endl;  //求关键字等于某值的元素个数
        pairs.insert(mmid::value_type(30,111.11));
        pairs.insert(mmid::value_type(10,22.22));
```

用小于号比较大小

相当于pair<int, double>(15, 2.7)，是一个临时对象

输出:

1) 0

2) 2

```
    pairs.insert(mmid::value_type(25,33.333));
    pairs.insert(mmid::value_type(20,9.3));
    for( mmid::const_iterator i = pairs.begin();
        i != pairs.end() ;i ++ )
        cout << "(" << i->first << "," << i->second   << ")" << ",";
}
```

输出：

1) 0

2) 2

(10,22.22),(15,2.7),(15,99.3),(20,9.3),(25,33.333),(30,111.11)

multimap例题

一个学生成绩录入和查询系统，
接受以下两种输入：

```
Add name id score
Query score
```

name是个字符串，中间没有空格，代表学生姓名。id是个整数，代表学号。score是个整数，表示分数。学号不会重复，分数和姓名都可能重复。

两种输入交替出现。第一种输入表示要添加一个学生的信息，碰到这种输入，就记下学生的姓名、id和分数。第二种输入表示要查询，碰到这种输入，就输出已有记录中分数比score低的最高分获得者的姓名、学号和分数。如果有多个学生都满足条件，就输出学号最大的那个学生的信息。如果找不到满足条件的学生，则输出“Nobody”

输入样例：

Add Jack 12 78
Query 78
Query 81
Add Percy 9 81
Add Marry 8 81
Query 82
Add Tom 11 79
Query 80
Query 81

添加记录和查询记录的时间复杂度为 log(n)

用关联容器中的multimap可以做到，而vector做不到

用分数作为关键字

输出果样例：

Nobody
Jack 12 78
Percy 9 81
Tom 11 79
Tom 11 79

```
#include <iostream>
#include <map> //使用multimap需要包含此头文件
#include <string>
using namespace std;
class CStudent
{
public:
                struct CInfo //类的内部还可以定义类
                {
                        int id;
                        string name;
                };
                int score;
                CInfo info; //学生的其他信息
};
typedef multimap<int, CStudent::CInfo> MAP_STD;
```

```
int main()    {
        MAP_STD mp;
        CStudent st;
        string cmd;
        while( cin >> cmd ) {
                if( cmd == "Add") {
                    cin >> st.info.name >> st.info.id >> st.score ;
                    mp.insert(MAP_STD::value_type(st.score,st.info ));
                }
                else if( cmd == "Query" ){
                        int score;
                        cin >> score;
                        MAP_STD::iterator p = mp.lower_bound (score);
                        if( p!= mp.begin()) {
                            --p;
                            score = p->first; //比要查询分数低的最高分
                            MAP_STD::iterator maxp = p;
                            int maxId = p->second.id;
```

pair模板的临时对象

```
int main()    {
        MAP_STD mp;
        CStudent st;
        string cmd;
        while( cin >> cmd ) {
                if( cmd == "Add") {
                    cin >> st.info.name >> st.info.id >> st.score ;
                    mp.insert(MAP_STD::value_type(st.score,st.info ));
                }
                else if( cmd == "Query" ){
                        int score;
                        cin >> score;
                        MAP_STD::iterator p = mp.lower_bound (score);
                        if( p!= mp.begin()) {
                            --p;
                            score = p->first; //比要查询分数低的最高分
                            MAP_STD::iterator maxp = p;
                            int maxId = p->second.id;
```

iterator lower_bound (const T & val);
查找一个最大的位置 it,使得 [begin(),it) 中所有元素的first 都比 val 小。

```
            for( ; p != mp.begin() && p->first ==
                        score; --p) {
                //遍历所有成绩和score相等的学生
                        if( p->second.id > maxId ) {
                                maxp = p;
                                maxId = p->second.id ;
                        }
            }
            if( p->first == score) {
        //如果上面循环是因为 p == mp.begin()
        // 而终止，则p指向的元素还要处理
                        if( p->second.id  > maxId ) {
                                maxp = p;
                                maxId = p->second.id ;
                        }
            }
```

```
                                    cout << maxp->second.name <<
                                     " “ << maxp->second.id << " "
                                         << maxp->first << endl;
                                }
                                else
//lower_bound的结果就是 begin，说明没人分数比查询分数低
                                    cout << "Nobody" << endl;
                            }
                    }
                    return 0;
}
```

```
                                        cout << maxp->second.name <<
                                         " “ << maxp->second.id << " "
                                             << maxp->first << endl;
                                }
                                else
//lower_bound的结果就是 begin，说明没人分数比查询分数低
                                        cout << "Nobody" << endl;
                        }
                }
                return 0;
}
```

```
 mp.insert(MAP_STD::value_type(st.score,st.info ));
//mp.insert(make_pair(st.score,st.info )); 也可以
```

make_pair是一个函数模板，返回一个pair模板类的对象，first和second和st.score, st.info的类型是相同的。

map

map 不允许里面有两个元素的first成员变量相同

```
template<class Key, class T, class Pred = less<Key>,
        class A = allocator<T> >
class map {
    ....
    typedef pair<const Key, T> value_type;
    .......
};
```

- map 中的元素都是pair模板类对象。关键字(first成员变量)各不相同。元素按照关键字从小到大排列，缺省情况下用 less<Key>,即“<” 定义“小于”。

map的[ ]成员函数

若pairs为map模版类的对象，

pairs[key]

返回对关键字等于key的元素的值(second成员变量)的引用。若没有关键字为key的元素，则会往pairs里插入一个关键字为key的元素，其值用无参构造函数初始化，并返回其值的引用.

map的[ ]成员函数

若pairs为map模版类的对象，

pairs[key]

返回对关键字等于key的元素的值(second成员变量）的引用。若没有关键字为key的元素，则会往pairs里插入一个关键字为key的元素，其值用无参构造函数初始化，并返回其值的引用.

如：

map<int,double> pairs;

则

pairs[50] = 5; 会修改pairs中关键字为50的元素，使其值变成5。

若不存在关键字等于50的元素，则插入此元素，并使其值变为5。

map示例

```
#include <iostream>
#include <map>
using namespace std;
template <class Key,class Value>
ostream & operator <<( ostream & o, const pair<Key,Value> & p)
{
    o << "(" << p.first  << "," << p.second << ")";
    return o;
}
```

```
int main()  {
    typedef map<int, double, less<int> > mmid;
    mmid pairs;
    cout << "1) " << pairs.count(15) << endl;
    pairs.insert(mmid::value_type(15,2.7));
    pairs.insert(make_pair(15,99.3)); //make_pair生成一个pair对象
    cout << "2) " << pairs.count(15) << endl;
    pairs.insert(mmid::value_type(20,9.3));
    mmid::iterator i;
    cout << "3) ";
    for( i = pairs.begin(); i != pairs.end();i ++ )
        cout << * i  << ",";
    cout << endl;
```

用小于号比大小

pair的模板类的对象

插入失败

输出：

1) 0

2) 1

3) (15,2.7),(20,9.3),

```
    cout << "4) ";
    int n =  pairs[40];//如果没有关键字为40的元素，则插入一个
    for( i = pairs.begin(); i != pairs.end();i ++ )
        cout << * i  << ",";
    cout << endl;
    cout << "5) ";
    pairs[15] = 6.28; //把关键字为15的元素值改成6.28
    for( i = pairs.begin(); i != pairs.end();i ++ )
        cout << * i  << ",";
}
```

默认为0

输出：

1) 0

2) 1

3) (15,2.7),(20,9.3),

4) (15,2.7),(20,9.3),(40,0),

5) (15,6.28),(20,9.3),(40,0),

In-Video Quiz

1. 下面三段程序，哪个不会导致编译出错?(提示，本题非常坑)

A)multimap <string,greater<string> > mp;

B)multimap <string,double,less<int> > mp1; mp1.insert(make_pair(“ok”,3.14));

C)multimap<string,double> mp2; mp2.insert(“ok”,3.14);

D)都会导致编译出错

2. 有对象map<string,int> mp; 则表达式mp[“ok”] 的返回值类型是：

A)Int B)int & C)string D)string &