



# 程序设计与算法(一)

## C语言程序设计

郭 炜

微信公众号



微博: <http://weibo.com/guoweiofpku>

**学会程序和算法，走遍天下都不怕!**

讲义照片均为郭炜拍摄



北京大学  
PEKING UNIVERSITY

信息科学技术学院

指定教材：

## 《新标准C++程序设计教程》

郭炜 编著

清华大学出版社

重点大学计算机专业系列教材

## 新标准C++程序设计教程

郭炜 编著



清华大学出版社



## STL 初步(二)



北京大学  
PEKING UNIVERSITY

信息科学技术学院

multimap



泰国普吉岛

# multimap的用法

multimap容器里的元素，都是pair形式的

```
multimap<T1, T2> mp;
```

则mp里的元素都是如下类型：

```
struct {  
    T1 first;    //关键字  
    T2 second;  //值  
};
```

multimap中的元素按照first排序，并可以按first进行查找

缺省的排序规则是 "a.first < b.first" 为true, 则a排在b前面

## multimap的应用

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是个不超过16字符的字符串，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

### 输出样例:

Nobody

Jack 12 78

Percy 9 81

Tom 11 79

Tom 11 79

```
#include <iostream>
#include <map> //使用multimap和map需要包含此头文件
```

```
#include <cstring>
```

```
using namespace std;
```

```
struct StudentInfo {
```

```
    int id;
```

```
    char name[20];
```

```
};
```

```
struct Student {
```

```
    int score;
```

```
    StudentInfo info;
```

```
};
```

```
typedef multimap<int,StudentInfo> MAP_STD;
```

```
// 此后 MAP_STD 等价于 multimap<int,StudentInfo>
```

```
// typedef int * PINT;
```

```
// 则此后 PINT 等价于 int *。 即 PINT p; 等价于 int * p;
```



```
int main()    {
    MAP_STD mp;
    Student st;
    char cmd[20];
    while( cin >> cmd ) {
        if( cmd[0] == 'A' )  {
            cin >> st.info.name >> st.info.id >> st.score ;
            mp.insert(make_pair(st.score,st.info ));
        } //make_pair生成一个 pair<int,StudentInfo>变量
           //其first 等于 st.score, second 等于 st.info
        else if(  cmd[0] == 'Q' ){
            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;  
    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;
    }
//lower_bound的结果就是 begin, 说明没人分数比查询分数低
    else cout << "Nobody" << endl;

}

return 0;

}

```



北京大学  
PEKING UNIVERSITY

信息科学技术学院

map



越南岬港

# map的用法

和multimap区别在于：

- 不能有关键字重复的元素
- 可以使用 `[]`，下标为关键字，返回值为`first`和关键字相同的元素的`second`
- 插入元素可能失败

```
#include <iostream>
#include <map>
#include <string>
using namespace std;
struct Student {
    string name;
    int score;
};
Student students[5] = {
    {"Jack",89}, {"Tom",74}, {"Cindy",87}, {"Alysa",87}, {"Micheal",98}};
typedef map<string,int> MP;
int main()
{
    MP mp;
    for(int i = 0;i < 5; ++i)
        mp.insert(make_pair(students[i].name,students[i].score));
    cout << mp["Jack"] << endl;    // 输出 89
    mp["Jack"] = 60; //修改名为"Jack"的元素的second
```

```
for(MP::iterator i = mp.begin(); i != mp.end(); ++i)
    cout << "(" << i->first << "," << i->second << ")" << " ";
```

**//输出:** (Alysa,87) (Cindy,87) (Jack,60) (Micheal,98) (Tom,74)

```
cout << endl;
```

```
Student st;
```

```
st.name = "Jack";
```

```
st.score = 99;
```

```
pair<MP::iterator, bool> p =
```

```
    mp.insert(make_pair(st.name,st.score));
```

```
if( p.second )
```

```
    cout << "(" << p.first->first << ","
```

```
    << p.first->second << ") inserted" << endl;
```

```
else
```

```
    cout << "insertion failed" << endl; //输出此信息
```

```
mp["Harry"] = 78; //插入一元素, 其first为"Harry", 然后将其second改为78
```

```
MP::iterator q = mp.find("Harry");
```

```
cout << "(" << q->first << "," << q->second << ")" << endl;
```

**//输出** (Harry,78)

```
return 0;
```

```
}
```

## map例题：单词词频统计程序

输入大量单词，每个单词，一行，不超过20字符，没有空格。按出现次数从多到少输出这些单词及其出现次数。出现次数相同的，字典序靠前的在前面

**输入样例：**

```
this  
is  
ok  
this  
plus  
that  
is  
plus  
plus
```

**输出样例：**

```
plus 3  
is 2  
this 2  
ok 1  
that 1
```



```
#include <iostream>
#include <set>
#include <map>
#include <string>
using namespace std;
struct Word {
    int times;
    string wd;
};
struct Rule {
    bool operator () ( const Word & w1,const Word & w2) const {
        if( w1.times != w2.times)
            return w1.times > w2.times;
        else
            return w1.wd < w2.wd;
    }
};
```

```
int main()
{
    string s;
    set<Word,Rule> st;
    map<string,int> mp;
    while( cin >> s )
        ++ mp[s] ;
    for( map<string,int>::iterator i = mp.begin();
        i != mp.end(); ++i) {
        Word tmp;
        tmp.wd = i->first;
        tmp.times = i->second;
        st.insert(tmp);
    }
    for(set<Word,Rule>::iterator i = st.begin();
        i != st.end(); ++i)
        cout << i->wd << " " << i->times << endl;
}
```