

# CS101

A Mars rover, likely a Curiosity rover, is shown on a rocky, reddish-brown landscape under a hazy, orange sky. The rover is positioned on the left side of the frame, facing right. It has six large, treaded wheels and a complex body with various instruments and cameras. A long shadow of the rover is cast onto the ground to its left. In the background, there are low, rolling hills under a bright, hazy sky.

信奥  
算法

课件下载地址:

<http://pan.baidu.com/s/1o885tz0>

作业网站:

<http://120.132.18.213:8080/thrall-web/main#home>

# 自定义类型的数据容器

multiset

set




struct

# 自定义数据类型和比较规则


运行程序“自定义数据类型和比较规则”

观察运行结果

思考语句的含义



```
9 int main() {
10     hero x1={70,"SUPERMAN"}; hero x2={70,"CAPTAINAMERICA"};
11     hero x3={100,"DORAEMON"}; hero x4={70,"SUPERMAN"};
12     cmp c,d;
13     cout<<c(x1,x2)<<" "<<c(x2,x1)<<endl;
14     cout<<c(x2,x3)<<" "<<c(x3,x2)<<endl;
15     cout<<c(x1,x4)<<" "<<c(x4,x1)<<endl;
16     cout<<d(x1,x4)<<" "<<d(x4,x1)<<endl;
17     return 0;
18 }
```



The code defines a `hero` struct with `id` and `name` fields. It creates four `hero` objects: `x1` (70, "SUPERMAN"), `x2` (70, "CAPTAINAMERICA"), `x3` (100, "DORAEMON"), and `x4` (70, "SUPERMAN"). The `cmp` function is used to compare `c` and `d` objects. The `cout` statements output the results of comparisons between `x1`, `x2`, `x3`, and `x4`. The `return 0;` statement indicates the end of the program.

# struct自定义类型

```
4 struct hero{int rp; string name;};//自定义hero类型
```

struct用来定义结构体，也就是新的类型

```
1 struct hero{  
2     string name;  
3     int life,speed;  
4 };
```

```
6 struct student{  
7     int id,score;  
8     string name;  
9 };
```

```
11 struct food{  
12     double price;  
13     int quantity;  
14 };
```

# struct自定义比较规则

```
4 struct cmp{//自定义cmp类型，可用括号()为两个hero比较
5     bool operator()(const hero& a,const hero& b)const{
6         return a.rp>b.rp||a.rp==b.rp&&a.name<b.name;
7     }
8 };
```

struct定义cmp类型里只有一个比较函数

rp越高排名越靠前，如果rp相等按照字典序排

operator()是括号操作符  
自定义为能比较两个hero的新操作

```
1 #include<iostream>
2 using namespace std;
3 struct hero{int rp; string name;}; //自定义hero类型
4 struct cmp{ //自定义cmp类型, 可用括号()为两个hero比较
5     bool operator()(const hero& a, const hero& b) const {
6         return a.rp>b.rp || a.rp==b.rp && a.name<b.name;
7     }
8 };
9 int main() {
10     hero x1={70, "SUPERMAN"}; hero x2={70, "CAPTAINAMERICA"};
11     hero x3={100, "DORAEMON"}; hero x4={70, "SUPERMAN"};
12     cmp c, d;
13     cout<<c(x1, x2)<<" "<<c(x2, x1)<<endl;
14     cout<<c(x2, x3)<<" "<<c(x3, x2)<<endl;
15     cout<<c(x1, x4)<<" "<<c(x4, x1)<<endl;
16     cout<<d(x1, x4)<<" "<<d(x4, x1)<<endl;
17     return 0;
18 }
```



# struct自定义顺序

```
4 struct cmp{//自定义cmp类型, 可用括号()为两个hero比较
5     bool operator()(const hero& a,const hero& b){
6         return a.rp>b.rp||a.rp==b.rp&&a.name<b.name;
7     }
8 };
```

struct cmp

可理解为自定义“法官”类型，或“裁判”

cmp c,d;

可理解为c,d为两位法官

c(x1,x2)

括号()可理解为法庭，  
()法庭的规则是自定义的  
c “法官” 判定是否x1胜过x2

d(x4,x1)

d “法官” 判定是否x4胜过x1



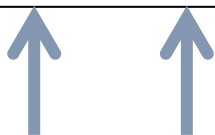
# 自定义类型multiset

运行程序“自定义类型multiset”

观察运行结果

思考multiset定义语句的含义

```
11 multiset<hero,cmp> s; //定义multiset和比较规则
12 multiset<hero,cmp>::iterator it; //定义迭代器
```



```
1 #include<iostream>
2 #include<set> //引入set库
3 using namespace std;
4 struct hero{int rp; string name;}; //自定义hero类型
5 struct cmp{//自定义cmp()比较规则
6     bool operator()(const hero& a,const hero& b)const{
7         return a.rp>b.rp||a.rp==b.rp&&a.name<b.name;
8     }
9 };
10 int main() {
11     multiset<hero,cmp> s; //定义multiset和比较规则
12     multiset<hero,cmp>::iterator it; //定义迭代器
13     hero x1={70,"SUPERMAN"};
14     hero x2={70,"CAPTAINAMERICA"};
15     hero x3={100,"DORAEMON"};
16     hero x4={70,"SUPERMAN"};
17     s.insert(x1);    s.insert(x2);
18     s.insert(x3);    s.insert(x4);
19     for(it=s.begin();it!=s.end();it++)
20         cout<<(*it).name<<" "<<(*it).rp<<endl;
21     return 0;
22 }
```

# 自定义类型set

运行程序“自定义类型set”

观察运行结果

思考：对于自定义类型，**set**是如何去重的

```
1 #include<iostream>
2 #include<set> //引入set库
3 using namespace std;
4 struct hero{int rp; string name;};
5 struct cmp{
6     bool operator()(const hero& a,const hero& b)const{
7         return a.rp>b.rp||a.rp==b.rp&&a.name<b.name;
8     }
9 };
10 int main() {
11     set<hero,cmp> s; //定义multiset和比较规则
12     set<hero,cmp>::iterator it; //定义迭代器
13     hero x1={70,"SUPERMAN"};
14     hero x2={70,"CAPTAINAMERICA"};
15     hero x3={100,"DORAEMON"};
16     hero x4={70,"SUPERMAN"};
17     s.insert(x1);    s.insert(x2);
18     s.insert(x3);    s.insert(x4);
19     for(it=s.begin();it!=s.end();it++)
20         cout<<(*it).name<<" "<<(*it).rp<<endl;
21     return 0;
22 }
```

# 如何判断相等？

```
cmp c;
```

//定义c为cmp比较类型，用于判断顺序

如果比较 $c(x1, x2)$ 返回1，那么x1排在x2前面

如果比较 $c(x2, x1)$ 返回1，那么x2排在x1前面

如果比较 $c(x1, x2)$ 返回0，并且 $c(x2, x1)$ 返回0  
那么x1和x2可以不分前后，认为相等

# 总结

```
#include<set>
```

```
struct hero{  
    string name;  
    int x;  
};
```

```
struct cmp{  
    bool operator()(const hero&a, const hero&b){  
        //规则细节  
    };  
};
```

```
multiset<hero,cmp> ms;  
multiset<hero,cmp>::iterator it;
```

第一步 引入set库

第二步 自定义数据类型

第三步 自定义比较规则

定义括号操作符()

第四步 定义multiset  
和迭代器

# 综合练习



# 整数奇偶排序

给定10个整数的序列，要求对其重新排序。排序要求：

- 1.奇数在前，偶数在后；
- 2.奇数按从大到小排序；
- 3.偶数按从小到大排序。

样例输入

4 7 3 13 11 12 0 47 34 98

样例输出

47 13 11 7 3 0 4 12 34 98

```
1 #include<iostream>
2 #include<set>
3 using namespace std;
4 struct cmp{
5     bool operator()(const int&a,const int&b)const{
6         if(a%2&&b%2==0) return 1;
7         if(a%2==0&&b%2) return 0;
8         if(a%2&&a>b) return 1;
9         if(a%2==0&&a<b) return 1;
10        return 0;
11    }
12 };
13 int main(){
14     multiset<int,cmp> ms;
15     multiset<int,cmp>::iterator it;
16     for(int i=0;i<10;i++) {
17         int x;
18         cin>>x; ms.insert(x);
19     }
20     for(it=ms.begin();it!=ms.end();it++)
21         cout<<*it<<" ";
22     return 0;
23 }
```

# 分数排序

将n个学生的分数进行从高到低的排序，**如果同分请按照输入顺序排序。**

输入第一行为n，之后每行是学生姓名和整数分数。

样例输入

5

John 59

Bob 59

Leo 60

Wang 100

Tom 100

样例输出

Wang Tom Leo John Bob

```
1 #include<iostream>
2 #include<string>
3 #include<set>
4 using namespace std;
5 struct student{
6     string name;
7     int id,s;
8 };
9 struct cmp{
10     bool operator()(const student&a,const student&b)const{
11         if(a.s>b.s) return 1;
12         if(a.s<b.s) return 0;
13         if(a.id<b.id) return 1;
14         return 0;
15     }
16 };
```

```
17 int main(){
18     multiset<student,cmp> ms;
19     multiset<student,cmp>::iterator it;
20     int n,i;
21     cin>>n;
22     for(i=0;i<n;i++) {
23         student x; x.id=i;
24         cin>>x.name>>x.s; ms.insert(x);
25     }
26     for(it=ms.begin();it!=ms.end();it++)
27         cout<<(*it).name<<' ';
28     return 0;
29 }
```

分配编号

---

易错点

# 因为const遗漏而出错

```
1  #include<iostream>
2  #include<string>
3  #include<set>
4  using namespace std;
5  struct dog{string name;int year;};
6  struct cmp{
7      bool operator()(const dog&a,const dog&b){//这行漏了const
8          if(a.name<b.name)return 1;
9          if(a.name>b.name)return 0;
10         if(a.year<b.year)return 1;
11         return 0;
12     }
13 };
14 int main() {
15     set<dog,cmp> s;
16     dog d={"john",2000};
17     s.insert(d);
18     cout<<s.count(d); //出错了
19     if(s.find(d)!=s.end())cout<<1; //这句没问题
20     return 0;
21 }
```



# 易错点汇总

打开“易错点汇总”程序

观察程序每一行，找到错误点，并修改

无法找出错误时，再试试运行

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4  struct student{
5      string name;
6      int id,s
7  }
8  struct cmp(
9      bool operator(const student&a, student b){
10         if(a.s>b.s) return 1;
11         if(a.s<b.s) return 0;
12         if(a.id<b.id) return 1;
13     }
14 )
15 )
```

```
16 int main(){
17     set<student> ms;
18     set<student>::iterator it;
19     int n,i;
20     cin>>n;
21     for(i=0;i<=n;i++) {
22         student x; x.id=i;
23         cin>>x.name>>x.s; ms.insert(x);
24     }
25     for(it=ms.begin();it<ms.end();it++)
26         cout<<*it.name<<' ';
27     return 0;
28 }
```

# 参考资料

---

<http://www.cplusplus.com/reference/set/set/>

<http://www.cplusplus.com/reference/set/multiset/>