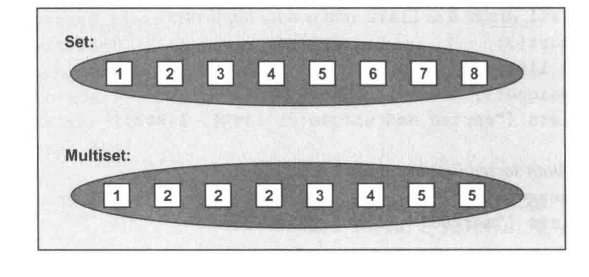
set或multiset的元素能够依据“排序准则”被比较。



排序准则：

1. 必须非对称：

对operator <而言，如果x < y为true，则y < x为false。注意：x<y为false，y<x不一定就为true，也可能为false，此时认为x=y。

1. 可传递：

如果x<y为true且y<z为true，x<z为true

1. 非自反：

x<x永远为false

1. 等效传递性：（主要针对multiset）

如果x=y且y=z，那么x=z。

!(x<y) && !(y<x)为true且!(y<z) && !(z<y)为true，则!(x<z) && !(z<x)

multiset的等效元素的次序是随机并稳定。

set和multiset通常以红黑树实现。

set和multiset不能直接修改元素，以避免破坏原本正确的顺序。set和multiset不提供任何操作函数可以直接访问元素，

set和multiset如果希望修改元素值，必须先删除元素，然后再插入新的元素。

例：程序stl\_test48：

*set*<int> coll1;

coll1.*insert*(1);

coll1.*insert*(2);

coll1.*insert*(4);

coll1.*insert*(5);

coll1.*insert*(6);

// 返回第1个可以安插值为3的元素的位置

*cout* << "lower\_bound(3): " << \*coll1.*lower\_bound*(3) << *endl*;

// 返回最后一个可以安插值为3的元素的位置

*cout* << "upper\_bound(3): " << \*coll1.*upper\_bound*(3) << *endl*;

// 返回第1个和最后1个可以安插值为3的元素的区间

*cout* << "equal\_range(3): " << \*coll1.*equal\_range*(3).*first*

<< " " << \*coll1.*equal\_range*(3).*second* << *endl*;

*cout* << *endl*;

*cout* << "lower\_bound(5): " << \*coll1.*lower\_bound*(5) << *endl*;

*cout* << "upper\_bound(5): " << \*coll1.*upper\_bound*(5) << *endl*;

*cout* << "equal\_range(5): " << \*coll1.*equal\_range*(5).*first*

<< " " << \*coll1.*equal\_range*(5).*second* << *endl*;

输出为：

lower\_bound(3): 4

upper\_bound(3): 4

equal\_range(3): 4 4

lower\_bound(5): 5

upper\_bound(5): 6

equal\_range(5): 5 6

例：程序stl\_test48：

// 从大到小排序

*set*<int, *greater*<int>> coll2;

coll2.*insert*({4, 3, 5, 1, 6, 2});

for (auto elem : coll2)

{

*cout* << elem << ' ';

}

*cout* << *endl*;

auto status = coll2.*insert*(4);

if (status.*second*)

{

*cout* << "4 inserted as element " << *distance*(coll2.*begin*(), status.*first*) + 1 << *endl*;

}

else {

*cout* << "4 already exists" << *endl*;

}

// 从小到大排序

*set*<int> coll3(coll2.*cbegin*(), coll2.*cend*());

*copy*(coll3.*cbegin*(), coll3.*cend*(), *ostream\_iterator*<int>(*cout*, " "));

*cout* << *endl*;

// 删除从开头到元素3的位置所有的元素

coll3.*erase*(coll3.*begin*(), coll3.*find*(3));

int num = coll3.*erase*(5);

*cout* << num << " element(s) removed" << *endl*;

*copy*(coll3.*cbegin*(), coll3.*cend*(), *ostream\_iterator*<int>(*cout*, " "));

*cout* << *endl*;

输出为：

6 5 4 3 2 1

4 already exists

1 2 3 4 5 6

1 element(s) removed

3 4 6