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

set和multiset

关联容器

set, multiset, map, multimap

- 内部元素有序排列，新元素插入的位置取决于它的值，查找速度快。
- 除了各容器都有的函数外，还支持以下成员函数：

find: 查找等于某个值 的元素(x小于y和y小于x同时不成立即为相等)
lower_bound : 查找某个下界
upper_bound : 查找某个上界
equal_range : 同时查找上界和下界
count :计算等于某个值的元素个数(x小于y和y小于x同时不成立即为相等)
insert: 用以插入一个元素或一个区间

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容器里放着的都是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
```

multiset

```
template<class Key, class Pred = less<Key>,
        class A = allocator<Key> >
class multiset { …… };
```

类型参数可以有缺省值

- Pred类型的变量决定了multiset 中的元素，“一个比另一个小”是怎么定义的。multiset运行过程中，比较两个元素x,y的大小的做法，就是生成一个 Pred类型的变量，假定为 op,若表达式op(x,y) 返回值为true,则 x比y小。

  Pred的缺省类型是 less<Key>。

op可以是函数指针 函数对象

multiset

```
template<class Key, class Pred = less<Key>,
        class A = allocator<Key> >
class multiset { …… };
```

如果没有指定比大小的规则，缺省则默认调用<号比较大小

- Pred类型的变量决定了multiset 中的元素，“一个比另一个小”是怎么定义的。multiset运行过程中，比较两个元素x,y的大小的做法，就是生成一个 Pred类型的变量，假定为 op,若表达式op(x,y) 返回值为true,则 x比y小。

  Pred的缺省类型是 less<Key>。

- less 模板的定义：

```
template<class T>
struct less : public binary_function<T, T, bool>
{ bool operator()(const T& x, const T& y) { return x < y ; } const;   };
//less模板是靠 < 来比较大小的
```

multiset的成员函数

iterator find(const T & val);
在容器中查找值为val的元素，返回其迭代器。如果找不到，返回end()。

iterator insert(const T & val); 将val插入到容器中并返回其迭代器。

void insert(iterator first,iterator last); 将区间[first,last)插入容器。

int count(const T & val); 统计有多少个元素的值和val相等。

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

iterator upper_bound(const T & val);
查找一个最小的位置 it,使得[it,end()) 中所有的元素都比 val 大。

multiset的成员函数

pair<iterator,iterator> equal_range(const T & val);
同时求得lower_bound和upper_bound。 first为lower_bound, second为upper_bound

iterator erase(iterator it);
删除it指向的元素，返回其后面的元素的迭代器(Visual studio 2010上如此，但是在C++标准和Dev C++中，返回值不是这样)。

multiset 的用法

```
#include <set>
using namespace std;
class A {  };
int main()  {
        multiset<A> a;
        a.insert( A());  //error
}
```

multiset 的用法

```
#include <set>
using namespace std;
class A { };
int main() {
        multiset<A> a;
        a.insert( A()); //error
}
```

multiset <A> a;

就等价于

multiset<A, less<A>> a;

插入元素时，multiset会将被插入元素和已有元素进行比较。由于less模板是用 < 进行比较的，所以,这都要求 A 的对象能用 < 比较，即适当重载了 <

multiset 的用法示例

```
#include <iostream>
#include <set>  //使用multiset须包含此文件
using namespace std;
template <class T>
void Print(T first, T last)
{    for(;first != last ; ++first) cout << * first << " ";
      cout << endl;
}
class A    {
 private:
       int n;
 public:
       A(int n   ) { n = n  ; }
  friend bool operator< ( const A & a1, const A & a2 )   { return a1.n < a2.n; }
  friend ostream &  operator<< ( ostream & o, const A & a2 )  { o << a2.n;    return o; }
  friend class MyLess;
};
```

非常经典：友元函数和友元类

```
struct MyLess   {
     bool operator()( const A & a1, const A & a2)
    //按个位数比大小
     {  return ( a1.n % 10 ) < (a2.n % 10);  }
};
typedef multiset<A> MSET1;    //MSET1用 "<"比较大小
typedef multiset<A,MyLess> MSET2;  //MSET2用 MyLess::operator()比较大小
int main()
{
            const int SIZE = 6;
            A a[SIZE] = { 4,22,19,8,33,40 };
            MSET1 m1;
            m1.insert(a,a+SIZE);
            m1.insert(22);
            cout << "1) " << m1.count(22) << endl;       //输出 1) 2
            cout << "2) "; Print(m1.begin(),m1.end());   //输出 2) 4 8 19 22 22 33 40
```

```
//m1元素：4 8 19 22 22 33 40
MSET1::iterator pp =  m1.find(19);
if( pp != m1.end() ) //条件为真说明找到
            cout << "found" << endl;
            //本行会被执行，输出 found
cout << "3) "; cout << * m1.lower_bound(22) << ","
      <<* m1.upper_bound(22)<< endl;
 //输出 3) 22,33
pp = m1.erase(m1.lower_bound(22),m1.upper_bound(22));
//pp指向被删元素的下一个元素
cout << "4) "; Print(m1.begin(),m1.end()); //输出 4) 4 8 19 33 40
cout << "5) "; cout << * pp << endl;          //输出 5) 33
MSET2 m2;    // m2里的元素按n的个位数从小到大排
m2.insert(a,a+SIZE);
cout << "6) "; Print(m2.begin(),m2.end()); //输出 6) 40 22 33 4 8 19
return 0;
}
```

```
//m1元素：4 8 19 22 22 33 40
MSET1::iterator pp =  m1.find(19);
if( pp != m1.end() ) //条件为真说明找到
            cout << "found" << endl;
            //本行会被执行，输出 found
cout << "3) "; cout << * m1.lower_bound(22) << ","
       <<* m1.upper_bound(22)<< endl;
 //输出 3) 22,33
pp = m1.erase(m1.lower_bound(22),m1.upper_bound(22));
//pp指向被删元素的下一个元素
cout << "4) "; Print(m1.begin(),m1.end()); //输出 4) 4 8 19 33 40
cout << "5) "; cout << * pp << endl;          //输出 5) 33
MSET2 m2;    // m2里的元素按n的个位数从小到大排
m2.insert(a,a+SIZE);
cout << "6) "; Print(m2.begin(),m2.end()); //输出 6) 40 22 33 4 8 19
return 0;
}
```

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

输出：

1) 2

2) 4 8 19 22 22 33 40

3) 22,33

4) 4 8 19 33 40

5) 33

6) 40 22 33 4 8 19

set

```
template<class Key, class Pred = less<Key>,
    class A = allocator<Key> >
    class set { … }
```

插入set中已有的元素时，忽略插入。

set用法示例

```
#include <iostream>
#include <set>
using namespace std;
int main()    {
        typedef set<int>::iterator IT;
        int a[5] = { 3,4,6,1,2 };
        set<int> st(a,a+5);    // st里是 1 2 3 4 6
        pair< IT,bool> result;
        result = st.insert(5);  // st变成 1 2 3 4 5 6
        if( result.second )     //插入成功则输出被插入元素
            cout << * result.first  << " inserted" << endl; //输出: 5 inserted
        if( st.insert(5).second )    cout << * result.first  << endl;
        else
            cout << * result.first << " already exists" << endl; //输出 5 already exists
        pair<IT,IT> bounds = st.equal_range(4);
        cout << * bounds.first << "," << * bounds.second ;     //输出：4,5
        return 0;
}
```

Set没有指明排序方式，按照默认的 < 排序

指向插入的元素

重复插入5，插入失败

first指向比元素小的上界，second指向比元素大的下界。（上界和下界是左闭右开的）

输出结果：

5 inserted
5 already exists
4,5

In-Video Quiz

1. 以下哪个对象定义语句是错的?

A)pair<int,int> a(3.4,5.5);

B)pair<string,int> b;

C)pair<string,int> k(pair<char*,double> (“this”,4.5));

D)pair<string,int> x(pair<double,int> b(3.4,100));

2. 要让下面一段程序能够编译通过，需要重载哪个运算符？

class A { };

multiset<A,greater<A> > b;

b.insert(A());

A)== B)= C)> D)<

In-Video Quiz

1. 以下哪个对象定义语句是错的?

A)pair<int,int> a(3.4,5.5);

B)pair<string,int> b;

C)pair<string,int> k(pair<char*,double> (“this”,4.5));

D)pair<string,int> x(pair<double,int> b(3.4,100));

2. 要让下面一段程序能够编译通过，需要重载哪个运算符？

class A { };

multiset<A,greater<A> > b;

b.insert(A());

A)== B)= C)> D)<

In-Video Quiz

3. 下面程序片段输出结果是：

```
int a[] = { 2,3,4,5,7,3};
multiset<int> mp(a,a+6);
cout << * mp.lower_bound(4);
```

A)2 B)3 C)4 D)5

4. set<double> 类的equal_range成员函数的返回值类型是：

A)void

B)pair<set<double>::iterator, set<double>::iterator>

C)pair<int,int>

D)pair<set<double>::iterator,bool>