Lecture #8
STL Containers (cont.)

chapter:11

Associative Containers

Associative containers are non-linear containers that can locate elements stored in the container quickly. Such containers can store sets of values or *key/value* pairs. The four associative containers are <u>set</u>, <u>multiset</u>, <u>map</u>, and <u>multimap</u>.

These containers provide fast storage and quick access to retrieve elements using *keys* (often called search *keys*). Elements in an associative container are sorted according to some sorting criterion. By default, the elements are sorted using the $\underline{<}$ operator.

Common Functions in Associative Containers

Functions	**Description**
find(key)	Returns an iterator that points to the element with the specified key in the container.
lower_bound(key)	Returns an iterator that points to the first element with the specified key in the container.
upper_bound(key)	Returns an iterator that points to the next element after the last element with the specified key in the container.
count(key)	Returns the number of occurrences of the element with the specified key in the container.

Associative Containers

The STL provides eight associative containers with either of specific features as:

1) The container is a set or a map

2) The container require unique key or allows multiple keys

3) The container stores elements in order or not

Pair of values in its simplest

Example1: Using pair

```
#include <utility>  //for pair
#include <vector>

pair< string,int > MyPair;

MyPair.first="Nayeb";
MyPair.second =1;

cout << MyPair.first << " associated with :" << MyPair.second <<endl;

vector< pair<string, int> > VectorOfPair;

VectorOfPair.push_back(MyPair);

MyPair.first="Maleki";
MyPair.second =2;
VectorOfPair.push_back(MyPair);
```

PairDemo

Associative Containers: map

Example1: Using map container: inserting values

```
1)  #include <map>


2)  map<int, string> myMap; //create a map called myMap

3)  myMap[100] = "John Smith";  // 1) Assignment using array index notation


4)  myMap.insert(make_pair(104, "Jeff Reed"));  // insert using make_pair():


5)  myMap.insert(std::pair<int, string>(1923, "David D.")); // insert using STL pair


6)  myMap.insert(map<int, string>::value_type(1023,"Jarl J.")); // insert using value_type
```

Associative Containers: map

Example2: Using map container: how to print out

```
map<int, string> myMap; //create a map called myMap

for(const auto& mapElement:myMap)

    cout << mapElement.first <<":" <<mapElement.second <<endl;



map<int, string>::iterator p;

for (p = myMap.begin(); p != myMap.end(); p++)
    cout << p->first << " " << p->second << endl;
```

Associative Containers: multimap

Example1: Using multimap container

```
multimap <string, int> MyMultiMap;

MyMultiMap.insert(make_pair("nayeb",1));
MyMultiMap.insert(make_pair("nayeb",1));

multimap<string,int>:: iterator it;
for (it=MyMultiMap.begin();it != MyMultiMap.end(); it++)
{
    cout << it->first << " : " <<it->second <<endl;
}
```

MultiMapDemo

```
map<string, int> :: iterator it;

for (it=MyMap.begin(); it!=MyMap.end(); it++)
{
    multiMap.insert(make_pair(it->second, it->first));
}
```

MapToMultiMapDemo

Associative Containers: set, multiset

Example1: Using set container

```
#include <set>

set <int> set1, set2;
multiset <int> multiset1, multiset2;

cout << "Set: " << endl;
set1.insert(1);
set1.insert(1);  //? 1 again in a set? What the error would be?
set1.insert(2);
set2.insert(30);

cout << "Multiset: " << endl;

multiset1.insert(1);
multiset1.insert(1);   //And now?
multiset1.insert(2);
multiset2.insert(30)
```

SetDemo

Putting all together with a struct

Inclass example

Thank you