STL Associative Containers

CPE 212 -- Lecture 19 continued

** Notes based on
The C++ Standard Library: A Tutorial and Reference, by Niicolai M. Josuttis

Associative Containers

- Sets
- Multisets
- Maps
- Multimaps

- Typically implemented using a balanced binary tree (given the complexity constraints of the associated operations)

Sets/Multisets

- Sorted storage of key values to provide logarithmic search performance
- No direct changing of an element value since this would destroy the sorted order
  - Must remove the old value
  - Then add the new value

Set versus Multiset

- Set
  - Duplicate keys not allowed and are ignored upon insert
- Multiset
  - Duplicate keys allowed

Selected Set/Multiset Operations - 1

- set<T> someSet;
  - Creates set with no elements
- set<T, Op> someSet;
  - Creates set with no elements that will be sorted by Op
- ~set <T>()
  - Destructor

- multiset<T> someMultiSet;
  - Creates multiset with no elements
- multiset<T, Op> someMultiSet;
  - Creates multiset with no elements that will be sorted by Op
- ~multiset <T>()
  - Destructor

Selected Set/Multiset Operations - 2

- size()
  - Number of elements currently stored
- empty()
  - Returns true if empty, false otherwise
- find(T keyValue)
  - Returns position of first element with key equal to keyValue
- count(T keyValue)
  - Returns number of elements with key equal to keyValue
- insert(T keyValue)
  - Inserts element with key equal to keyValue, returning insert position
- erase(T keyValue)
  - Removes all elements with key equal to keyValue, returning number of elements removed
- clear()
  - Empties the container

Maps/Multimaps

- Sorted storage of (key, value) pairs
- Maps require unique keys
- Multimaps permit duplicate keys

Selected Map/Multimap Operations - 1

- map<T> someMap;
  - Creates map with no elements
- map <T, Op> someMap;
  - Creates map with no elements that will be sorted by Op
- ~map <T>()
  - Destructor

- multimap<T> someMultiMap;
  - Creates multimap with no elements
- multimap<T, Op> someMultiMap;
  - Creates multimap with no elements that will be sorted by Op
- ~multimap <T>()
  - Destructor

Selected Map/Multimap Operations - 2

- size()
  - Number of elements currently stored
- empty()
  - Returns true if empty, false otherwise
- find(T keyValue)
  - Returns position of first element with key equal to keyValue
- count(T keyValue)
  - Returns number of elements with key equal to keyValue
- insert(T keyValue)
  - Inserts element with key equal to keyValue, returning insert position
- erase(T keyValue)
  - Removes all elements with key equal to keyValue, returning number of elements removed
- clear()
  - Empties the container
- someMap[someKey] -- not for multimaps!!
  - Returns value associated with someKey or allows one to insert a value with key someKey

```
// Multimap Example1
#include <iostream>
#include <map>
#include <string>
#include <iomanip>
using namespace std;

int main()
{
  multimap<string, int>  exam;

  cout << endl;
  cout << "Number of elements = " << exam.size() << endl << endl;
  cout << "Insert <Smith, 95>" << endl;
  exam.insert(multimap<string,int>::value_type("Smith", 95));
  cout << "Insert <Jones, 78>" << endl;
  exam.insert(multimap<string,int>::value_type("Jones", 78));
  cout << "Insert <Smith, 84>" << endl << endl;
  exam.insert(multimap<string,int>::value_type("Smith", 84));


  cout << "Number of elements = " << exam.size() << endl;
  cout << "Number of elements with key 'Smith' = " << exam.count("Smith") << endl;


  cout << endl << "----------------------" << endl;
  cout << "Table of Values" << endl << endl;
  cout << setw(6) << "Key" << setw(10) << "Value" << endl;
  cout << "----------------------" << endl;
  multimap<string,int>::iterator  k;
  for(k = exam.begin(); k != exam.end(); k++)
    cout << setw(4)<< k->first << setw(10) << k->second << endl;
  cout <<endl;

  return 0;
} // End main()
```

```
-bash-3.2$ g++ mmap1.cpp
-bash-3.2$ ./a.out

Number of elements = 0

Insert <Smith, 95>
Insert <Jones, 78>
Insert <Smith, 84>

Number of elements = 3
Number of elements with key 'Smith' = 2

----------------------
Table of Values

   Key     Value
----------------------
Jones         78
Smith         95
Smith         84

-bash-3.2$
```

```
// Multimap Example2
#include <iostream>
#include <map>
#include <string>
#include <iomanip>
using namespace std;

typedef multimap<string, int> mmap;

int main()
{
  mmap  exam;

  cout << endl;
  cout << "Number of elements = " << exam.size() << endl << endl;
  cout << "Insert <Smith, 95>" << endl;
  exam.insert(mmap::value_type("Smith", 95));
  cout << "Insert <Jones, 78>" << endl;
  exam.insert(mmap::value_type("Jones", 78));
  cout << "Insert <Smith, 84>" << endl << endl;
  exam.insert(mmap::value_type("Smith", 84));


  cout << "Number of elements = " << exam.size() << endl;
  cout << "Number of elements with key 'Smith' = " << exam.count("Smith") << endl;


  cout << endl << "----------------------" << endl;
  cout << "Table of Values" << endl << endl;
  cout << setw(6) << "Key" << setw(10) << "Value" << endl;
  cout << "----------------------" << endl;
  mmap::iterator  k;
  for(k = exam.begin(); k != exam.end(); k++)
    cout << setw(4)<< k->first << setw(10) << k->second << endl;
  cout <<endl;

  return 0;
} // End main()
```

```
-bash-3.2$ g++ mmap2.cpp
-bash-3.2$ ./a.out

Number of elements = 0

Insert <Smith, 95>
Insert <Jones, 78>
Insert <Smith, 84>

Number of elements = 3
Number of elements with key 'Smith' = 2

----------------------
Table of Values

   Key     Value
----------------------
Jones         78
Smith         95
Smith         84

-bash-3.2$
```

```
// Map Example1
#include <iostream>
#include <map>
#include <string>
#include <iomanip>
using namespace std;

typedef map<string, int> mymap;

int main()
{
  mymap  exam;

  cout << endl;
  cout << "Number of elements = " << exam.size() << endl << endl;
  cout << "Insert <Smith, 95>" << endl;
  exam["Smith"] = 95;
  cout << "Insert <Jones, 78>" << endl;
  exam["Jones"] = 78;
  cout << "Insert <Smith, 84>" << endl << endl;
  exam["Smith"] = 84;

  cout << "Number of elements = " << exam.size() << endl;
  cout << "Number of elements with key 'Smith' = " << exam.count("Smith") << endl;

  cout << endl << "----------------------" << endl;
  cout << "Table of Values" << endl << endl;
  cout << setw(6) << "Key" << setw(10) << "Value" << endl;
  cout << "----------------------" << endl;
  mymap::iterator  k;
  for(k = exam.begin(); k != exam.end(); k++)
    cout << setw(4)<< k->first << setw(10) << k->second << endl;
  cout <<endl;

  return 0;
} // End main()
```

```
-bash-3.2$ g++ map1.cpp
-bash-3.2$ ./a.out

Number of elements = 0

Insert <Smith, 95>
Insert <Jones, 78>
Insert <Smith, 84>

Number of elements = 2
Number of elements with key 'Smith' = 1

----------------------
Table of Values

   Key     Value
----------------------
Jones         78
Smith         84


-bash-3.2$
```