C++ STL: sets and maps

Data and File Structures Laboratory

http://www.isical.ac.in/~dfslab/2017/index.html

C++ containers, sets, maps

- *Container:* data type for operating on a group of elements
  Example: array
- *Sets:* container for *distinct*, ordered data stored in a balanced binary tree structure supporting fast search
- *Maps: associative* container for $\langle$ *key*, *value* $\rangle$ pairs (where keys are *distinct* and *ordered*), stored in a balanced binary tree structure

Supported operations on sets, maps

- Add / remove element (if not already in the container)
- Get count of elements
- Check membership

Addition, removal, membership check guaranteed to take $O(\log N)$ time

Programs using sets

```
set<int> s;

for(int i = 1; i <= 100; i++) {
    s.insert(i); // Insert 100 elements, [1..100]
}

s.insert(42); // does nothing, 42 already exists in set

for(int i = 2; i <= 100; i += 2) {
    s.erase(i); // Erase even values
}

int n = int(s.size()); // n will be 50

if(s.find(42) != s.end()) {
    // 42 presents in set
}
else {
    // 42 not presents in set
}
```

Example program A – I

```
/* Source: http://www.yolinux.com/TUTORIALS/CppStlMultiMap.html */
#include <string.h>
#include <iostream>
#include <map>
#include <utility>

using namespace std;

int main()
{
    map<int, string> Employees;

    /* Note the use of array index notation */
    Employees[5234] = "Mike C.";
    Employees[3374] = "Charlie M.";
    Employees[1923] = "David D.";
    Employees[7582] = "John A.";
    Employees[5328] = "Peter Q.";

    cout << "Employees[3374]=" << Employees[3374] << endl << endl;
    cout << "Map size: " << Employees.size() << endl;
```

Example program A – II

```
    cout << endl << "Natural Order:" << endl;
    for( map<int,string>::iterator ii=Employees.begin(); ii!=Employees.end();
         ++ii)
    {
        cout << (*ii).first << ": " << (*ii).second << endl;
    }

    cout << endl << "Reverse Order:" << endl;
    for( map<int,string>::reverse_iterator ii=Employees.rbegin();
         ii!=Employees.rend(); ++ii)
    {
        cout << (*ii).first << ": " << (*ii).second << endl;
    }

    return 0;
}
```

Example program B – I

```
#include <string.h>
#include <iostream>
#include <map>
#include <utility>

using namespace std;

int main()
{
    map<string, int> Employees;

    // Assignment using array index notation
    Employees["Mike C."] = 5234;
    Employees["Charlie M."] = 3374;

    // Assignment using member function insert() and STL pair
    Employees.insert(std::pair<string,int>("David D.",1923));

    // 3) Assignment using member function insert() and "value_type()"
    Employees.insert(map<string,int>::value_type("John A.",7582));
```

Example program B – II

```
    // 4) Assignment using member function insert() and "make_pair()"
    Employees.insert(std::make_pair("Peter Q.",5328));

    cout << "Map size: " << Employees.size() << endl;
    for( map<string, int>::iterator ii=Employees.begin(); ii!=Employees.end();
         ++ii)
    {
        cout << (*ii).first << ": " << (*ii).second << endl;
    }

    return 0;
}
```

Example program C – I

```
#include <string.h>
#include <iostream>
#include <map>
#include <utility>

using namespace std;

struct ltstr {
    bool operator()(const char* s1, const char* s2) const
        {
            return strcmp(s1, s2) < 0;
        }
};


int main()
{
    map<const char*, int, ltstr> months;

    months["january"] = 31;
    months["february"] = 28;
```

Example program C – II

```
    months["march"] = 31;
    months["april"] = 30;
    months["may"] = 31;
    months["june"] = 30;
    months["july"] = 31;
    months["august"] = 31;
    months["september"] = 30;
    months["october"] = 31;
    months["november"] = 30;
    months["december"] = 31;

    cout << "june -> " << months["june"] << endl;
    map<const char*, int, ltstr>::iterator cur = months.find("june");
    map<const char*, int, ltstr>::iterator prev = cur;
    map<const char*, int, ltstr>::iterator next = cur;
    ++next;
    --prev;
    cout << "Previous (in alphabetical order) is " << (*prev).first << endl;
    cout << "Next (in alphabetical order) is " << (*next).first << endl;
}
```

References

- STL basics: https://www.topcoder.com/community/data-science/data-science-tutorials/power-up-c-with-the-standard-template-library-part-1
- Maps:
  - https://www.cprogramming.com/tutorial/stl/stlmap.html
  - http://devdocs.io/cpp/container/map
  - https://www.sgi.com/tech/stl/Map.html
  - http://www.yolinux.com/TUTORIALS/CppStlMultiMap.html