CSCI 262 Lectures 11 & 12 – Sets & Maps

Outline

- Sets – a container that holds *unique* elements
  - Principal operations are finding an element (testing to see if it is in the set), adding an element, and removing an element (we sometimes call these elements *keys*)
  - We also care about iterating over the contents of a set – *which we cannot do using indexing!*
- Maps – a container that associates *unique keys* with *values*
  - Principal operations including finding a *key* (is the key in the map), getting the value associated with a key, adding key/value pairs, removing key/value pairs, and updating the value associated with a key
  - We also care about iterating over the contents of a map – again, no indexing!
- Sets and maps may be available in ordered and unordered varieties, depending on the underlying data structure (hashtables for unordered, binary search trees for ordered)
  - Ordered is slightly slower, but allows iteration of elements in sort order
  - Unordered is slightly faster, but has no obvious ordering of the keys
- Iterators are objects which act like pointers into a collection, and can be used to iterate over sets and maps
  - This also gives us the range-based for loop
  - Iterators on sets let us look at the elements (sometimes called keys)
  - Iterators on maps let us look at the key-value *pairs*
- Map operations in C++ can be tricky – be careful to understand the different behaviors, especially of the [ ] operator

Readings

Read chapter 15.3 (on hashtables) for Monday; we'll look at binary search trees at a later time

Self Check

1. What is the easiest method to test to see if a key is already in a set or map?
2. What are the contents of a set of integers after inserting the values 42, 17, -3, 17, 8 ?
3. What does a C++ `map<string, string>` contain if we use insert or emplace to add the pairs { "dog", "bark" }, { "cat", "meow" }, { "dog", "woof" }, { "snake", "hiss" } ?
4. How can you update the value associated with a key in a map?
5. How can you print out all of the contents of a map?
6. What are the Big O complexities of all operations on sets and maps (answer for both ordered and unordered)?

For Further Practice

In lecture 11, we discussed ways to use a vector to implement a set data structure (not efficiently, but correctly). How might you implement a map using two vectors? How about one vector?

(Ordered) Sets

What is the output of the below code snippet?

```
#include <iostream>
#include <set>
using namespace std;

int main() {
    set<int> s;
    s.insert(1);
    s.insert(1);
    s.insert(2);
    s.insert(2);
    s.insert(0);
    s.insert(0);
    for (int i : s) {
        cout << i << " ";
    }
    return 0;
}
```

Unordered Sets

What is contained in the unordered set after the below code snippet?

```
#include <iostream>
#include <unordered_set>
using namespace std;

int main() {
    unordered_set<int> s;
    s.insert(1);
    s.insert(1);
    s.insert(2);
    s.insert(2);
    s.insert(0);
    s.insert(0);
    return 0;
}
```