STL Cheat Sheet 2 – set, map, stringstreams

Set/Map creation (#include <set>/#include <map>)

- Make an empty set of integers.

```
set<int> intSet;
```

- Make a set of integers containing the given array of numbers.

```
int array[] = {10, 20, 30, 40};
set<int> intSet(array, array + 4);
```

- Make an empty map from string to int.

```
map<string, int> siMap;
```

- Make an empty map from C-string to int.

```
struct compareString {
    bool operator()(const char *a, const char *b) const { return strcmp(a, b) < 0; }
};
map<const char *, int, compareString> siMap;
```

- Declare an iterator for a set of integers; declare an iterator for a string-to-int map (a map iterator represents a pair of key and value).

```
set<int>::iterator iSetItr;
```

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

Set/Map access and modification

- Number of items in a set (also for map).

```
intSet.size();
```

- Get an iterator which points to the beginning of the set.

```
iSetItr = intSet.begin();
```

- Get an iterator which points to the end of the map (one past the last element).

```
siMapItr = siMap.end();
```

- Get the value that is pointed to by the set iterator.

```
*iSetItr
```

- Get the key that is pointed to by the map iterator.

```
siMapItr->first
```

- Get the value that is pointed to by the map iterator.

```
siMapItr->second
```

Finding in Set/Map

- Find an item in a set (returns an iterator).
  `intSet.find(3)`
- See if an item is in a set.
  `if (intSet.find(3) != intSet.end()) ...`
- Find an item in a map (returns an iterator).
  `siMap.find("hello")`
- See if an item is in a set.
  `if (siMap.find("hello") != siMap.end()) ...`

Set/Map insertion and removal

- Place an item in a set.
  `intSet.insert(3)`
- Place a key/value in a map.
  `siMap["hello"] = 3`
- Removing an item from a set.
  `intSet.erase(intSet.find(3));` `intSet.erase(intSet.begin())`
- Removing an item from a map.
  `siMap.erase(siMap.find("hello"));` `siMap.erase(siMap.begin())`
- Clearing a set or a map.
  `intSet.clear(), siMap.clear()`

Getline and String streams (`#include <sstream>`)

- Getline can be used to grab a line of input at a time – everything until the next newline.
  `string line; getline(cin, line);`
- Using an input string stream to parse the words obtained by getline:

```
string line, token;
while (getline(cin, line)) {
    istringstream in(line);
    while (in >> token) { ...
```

- Beware of the unintended effects of the following code! If there is a newline immediately following the extracted integer, then the call to `getline` will obtain just an empty string.

```
int n; string line; cin >> n; getline(cin, line);
```

- Constructing strings with `ostringstream`:

```
ostringstream out;
out << 3 << " is less than " << 3.14159 << endl;
cout << out.str();
```