ITP 365

Sets and Maps

Lecture 5 Notes

1/24/17

**Sets:**

* Set is a collection of distinct objects
  + P = {2, 3, 5, 7, 11}
* Empty set is a set that contains nothing
  + O = {}
* Union between sets A and B = A U B -> contains all objects that are in either A or B
  + P = {2, 3, 5, 7, 11}
  + C = {4, 6, 8, 9, 10}
  + P U C = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
* Intersection
  + Set that contains all objects that are in BOTH A and B
* Subset
  + A is a subset of B if all the elements in A are also in B
  + C is a subset of C++ -> anything you can do in C you can do in C++ but doesn’t work the other way around
* Stanford C++ set -> **DON’T ALLOW RANDOM ACCESS LIKE ARRAYS, CAN ONLY SEE WHETHER IN OR OUT OF SET**
  + #include “set.h”
  + Set<int> mySet;
* Set member functions
  + Add – add element to set
    - mySet.add(element)
    - **IMPORTANT: Stored in ascending alphabetical order IN STANFORD SETS**
  + Contains
    - Check if set contains requested value -> if so, returns true
  + Remove
    - mySet.remove(element)
  + Union
    - Use the + operator to perform a union between two sets
    - mySet = set1 + set2
  + Intersection
    - Use the \* operator to perform a intersection between two sets
    - mySet = set1 \* set2
  + isSubsetOf
    - mySet.isSubsetOf(set2) -> retrusn true if set is a subset of the set passed as a parameter

**Maps:**

* Type of collection allows you to associate **a unique key with a value**
* Also called associative arrays (known as dictionary in python)
* Stanford C++ Map
  + #include “map.h”
  + **Map<int, std::string> myMap;**
    - **Keys are integers, and values are strings**
* Map member functions
  + Put – associates specified key with a value
    - myMap.put(key, value)
    - Ex. Map<int, std::string> months;
      * Months.put(1, “January”)
      * Months.put(2, “February”)
  + ContainsKey
    - myMap.containsKey(key) -> returns true if key is associated with a value in map
  + Get – returns the value associated with the key, provided that key is in Map
    - String value = myMap.get(key)
  + [] operator
    - Instead of put/get you can use []
    - Ex. Map<int, std::string> months
    - months[1] = “January”
    - months[2] = “February”
  + Remove
    - myMap.remove(key) -> removes key and associated value
* Range-based For loop – used to iterate over sets
  + For (type i : collection)
    - Visits each element in a collection
  + Modification in range-based
    - For (int& number : fibNumbers
      * Number \*= 2
  + If you use range-based for loops to iterate maps, you will only get keys