Maps

CSE 2020 Computer Science II

Learning Objectives

- Define Map abstract data type
- Design and implement Map ADT
- Apply map class template defined in STL

Map ADT

- A map is a container that stores a collection of ordered entries or pairs, each entry or pair consists of a key and a value, <key, value>.
  - Keys must be unique
  - Values need not be unique, so several keys can map to the same values
  - The entries/pairs in the map are sorted based on keys following a specific *ordering* criterion indicated by the internal comparable objects.
- Examples
  - (1001, "Bob"), (1002, "Mary"), (1003, "Bob"), ……
  - ("apple", "a round fruit of tree …"), ("bee", "an insect of …"), ("cat", "a small domesticated mammal")

Operations of Maps

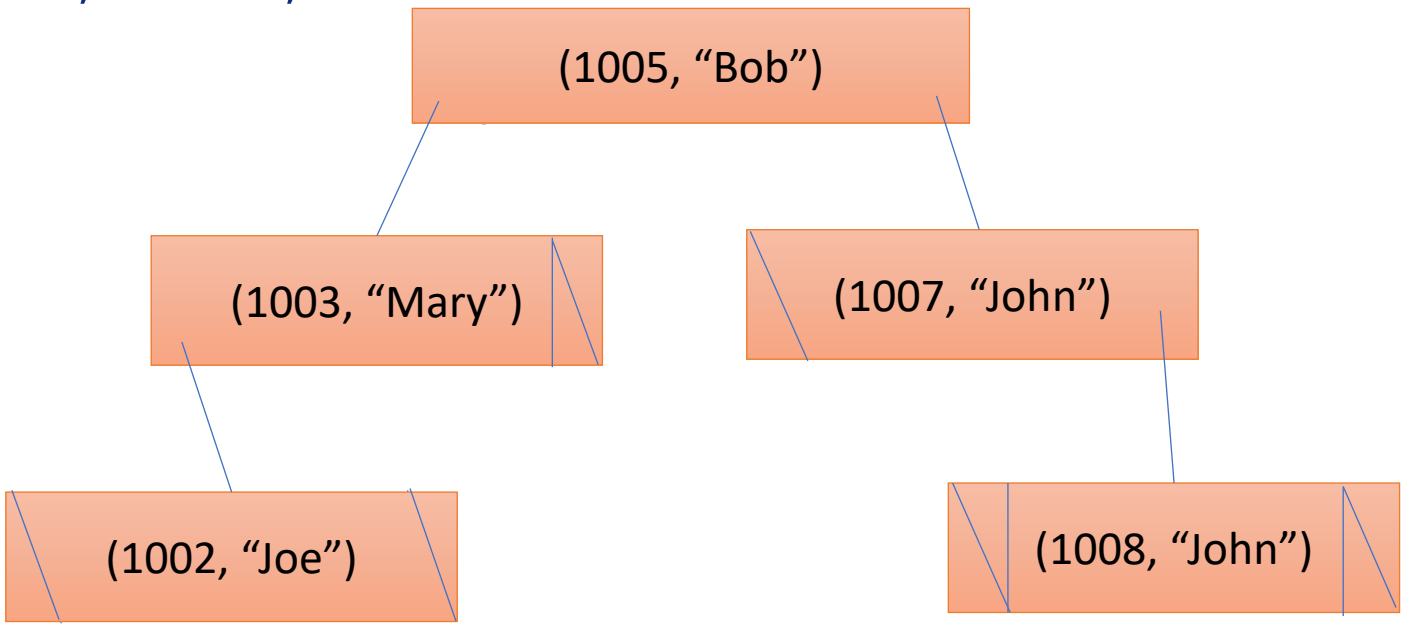
- bool **isEmpty**() const: returns true if map is empty
- int **getSize**() const: returns the number of entries in map
- V **operator []** (K key):
  - if key is not in map, inserts the entry with key returns default value
  - if key is in map, returns the value corresponding to key
- void **remove**(const K & key): removes the entry with the key from the map
- void **makeEmpty**(): makes the map to empty state

Example

- A collection of employee id and name entries <id, name>, such as <1005, “Bob”>, <1003, “Mary”>, <1002, “Joe”>, <1007, “John”>, <1008, “John”>
- employees is a map object
- insert <1005, “Bob”>, employees[1005] = “Bob”;
  - employees[1005] inserts <1005, “”>
  - assign “Bob” to the value of key 1005
- cout << employees[1005]; // Bob
- employees.remove(1005);

Using BST/Set Implement Map

- <1005, “Bob”>, <1003, “Mary”>, <1002, “Joe”>, <1007, “John”>, <1008, “John”>



- employees[1006] = “Mike”;
  - insert <1006, “”>, then update “” as “Mike”

Implementation

- storing entries in a set, or using Set class to implement Map class template
  - Pair.cpp, Pair class template has attributes key of type K and value of type V, (key, value), all comparisons are all based on keys

```
template <typename K, typename V>
class Pair{ …… };
```

  - MapSet.cpp, Set class template, iterator constructor and insert

```
iterator(BinaryNode*  p) : current(p) {  }
public iterator insert(const C & x)
private iterator insert(const C & x, BinaryNode* & t )
```

  - Map.cpp, Map is a set of entries/pairs,

```
private attribute is Set<Pair<K, V> > themap
V & operator [](K key)
void remove(K & key)
```

Implementation

- The code files contained in Map.txt are on Canvas
  - Pair.cpp
  - MapSet.cpp
  - Map.cpp
  - TestMap.cpp

map in STL

- In STL, map class template uses binary search tree/set to store a collection of entries.

```
#include <map>
#include <iterator>
map<int, string> mymap;
mymap[1005] = “bob”;
mymap[1002] = “mary”;
mymap[1008] = “joe”;
cout << mymap[1002];
mymap.erase(1002);
```