Advanced C

Associative Containers in C++

Th.S Nguyen Minh Anh

Phenikaa University

Last Update: 23rd February 2023

Recap: Vector

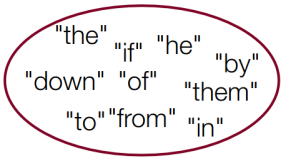
- Vector is good, and flexible
- **Fast** random access but **slow** retrieval

What if

Sets and Maps

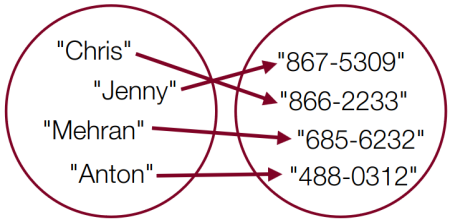
Sets

- Collection of elements with no duplicates



Maps

- Collection of key/value pairs
- The key is used to find its associated value

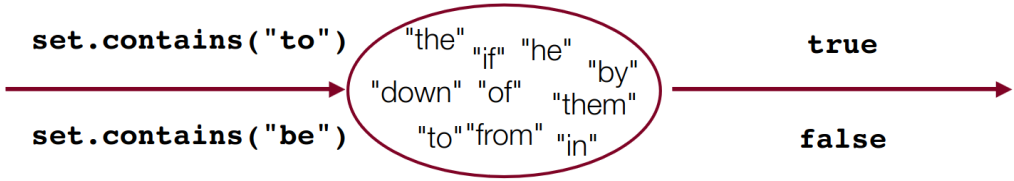


Outline

Set

Set

- A collection of elements with no duplicates.
- Operations include **`add`**, **`contains`**, and **`remove`**, and they are all fast
- Sets do not have indexes



Sets: Simple Example

```
Set<string> friends;
friends.add("chris");
friends.add("anton");
cout << friends.contains("voldemort") << endl;
for(string person : friends) {
   cout << person << endl;
}
```

Set Essentials

- **`int set.size()`**
  Returns the number of elements in the set.
- **`void set.add(value)`**
  Adds the new value to the set (ignores it if the value is already in the set)
- **`bool set.contains(value)`**
  Returns `true` if the value is in the set, `false` otherwise.
- **`void set.remove(value)`**
  Removes the value if present in the set. Does not return the value.
- **`bool set.isEmpty()`**
  Returns `true` if the set is empty, `false` otherwise.

Sets also have other helpful methods. See the online docs for more.

Looping Over a Set

```
for(type currElem : set) {
   // process elements one at a time
}
```

can't use a normal **for** loop and get each element[i]

```
for(int i=0; i < set.size(); i++) {
   // does not work, no index!
   cout << set[i];
}
```

Types of Sets

Set

- Iterate over elements in **sorted** order
- `O(log n) per retrieval`
- Implemented using a "binary search tree"

REALLY FAST!

HashSet

- Iterate over elements in **unsorted** order
- `O(1) per retrieval`
- Implemented using a "hash table"

REALLY,
RIDICULOUSLY FAST!

Set Operands

Sets can be compared, combined, etc.

- `s1 == s2`
  true if the sets contain exactly the same elements
- `s1 != s2`
  true if the sets don't contain the same elements
- `s1 + s2`
  returns the union of s1 and s2 (all elements in both)
- `s1 * s2`
  returns intersection of s1 and s2 (elements must be)
- `s1 - s2`
  returns difference of s1, s2 (elements in s1 but not s2)

Exercise

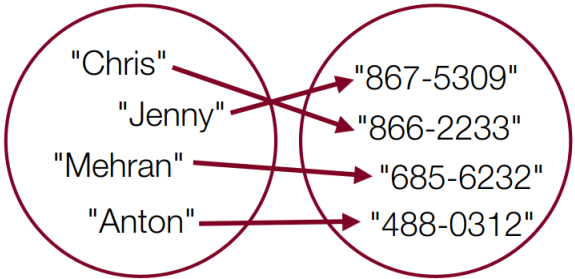
Count Unique Words

Outline

Maps

Map

- A collection of pairs `(k, v)`, sometimes called **key/value** pairs, where `v` can be found quickly if you know `k`.
- a.k.a. dictionary, associative array, hash
- A generalization of an array, where the "indexes" need not be ints.



Using Maps

A map allows you to get from one half of a pair to the other.

- **Store** an association from "Jenny" to "867-5309"
- **Get** Jenny's number

Maps are Everywhere

- Wiki: key = title, value = article
- Dictionary: key = word, value = meaning

Creating Maps

Requires 2 type parameters: one for keys, one for values.

```
// maps from string keys to integer values
Map<string, int> votes;
// maps from double keys to Vector<int> values
Map<string, Vector<string>> friendMap;
```

Map Methods

- `m.clear()`
- `m.contains()`
- `m[key]`
- `m.empty()`
- `m[key] = value` or `m.insert(pair(key,value))`
- `m.erase(key)`
- `m.size()`

Map Example

Types of Maps

Map

- Iterate over elements in **sorted** order
- `O(log n) per retrieval`
- Implemented using a "binary search tree"

REALLY FAST!

HashMap

- Iterate over elements in **unsorted** order
- `O(1) per retrieval`
- Implemented using a "hash table"

REALLY,
RIDICULOUSLY FAST!

Map Example: Tallying Votes

Tallying Words

Looping Over a Map

```
Map<string, double> gpa = load();
for (string name : gpa) {
   cout << name << "'s GPA is ";
   cout << gpa[name] << endl;
}
```

*The order is unpredictable in a HashMap

Outline

Multi what?