Programming Abstractions

CS106B

Cynthia Lee

Today’s Topics

ADTs

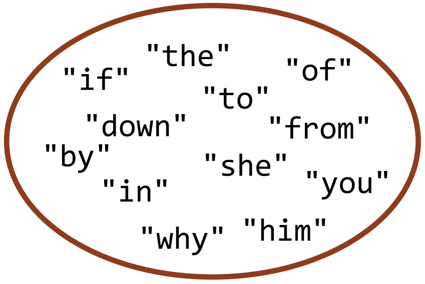
- Map
  - Example: counting words in text
- Containers within containers
  - Example: reference tests
  - Example: anagram finder (Friday)

Maps

What are they?

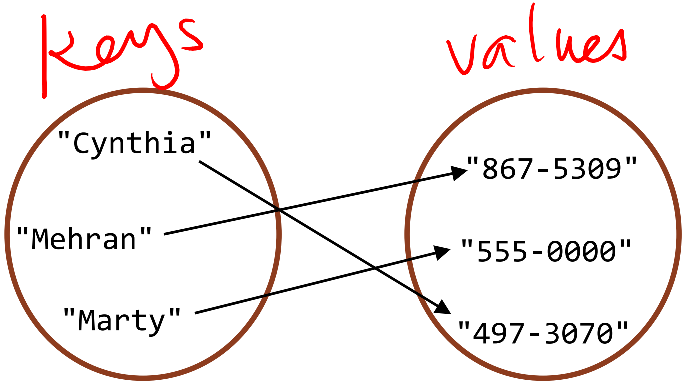
Associative containers

- Map
- Set
- Lexicon



"the" "if" "of" "to" "down" "from" "by" "she" "you" "in" "why" "him"

set



"Cynthia"
"Mehran"
"Marty"

"867-5309"
"555-0000"
"497-3070"

map

Not as concerned with order but with matching

- Set: associates **keys** with **membership** (yes or no)
- Map: associates **keys** with **values** (could be any type)

Stanford library Map *(selected member functions)*

```
template <typename KeyType, typename ValueType> class Map {
public:
    void add(const KeyType& key, const ValueType& value);

    bool containsKey(const KeyType& key) const;

    ValueType get(const KeyType& key) const;

    ValueType operator [](const KeyType& key) const;
...
}
```

- `Map<string, string> phone;        // Map takes *two* template parameters`
- `phone["Cynthia"] = "497-3070";     // two options for add syntax`
- `phone.add("Mehran", "867-5309");   // two options for add syntax`
- `cout << phone["Cynthia"] << endl;      // two options for get syntax`
- `cout << phone.get("Mehran") << endl;   // two options for get syntax`

Map programming exercise

Write a program to count <u>the number of occurrences</u> of each unique word in a text file (e.g. *Poker* by Zora Neale Hurston).

- **First do an initial report:**
  - Print all words that appeared in the book at least 10 times, in alphabetical order
- **Then go into interactive query mode:**
  - The user types a word and we report *how many times* that word appeared in the book (repeat in a loop until quit).

Map programming exercise

Write a program to count the number of occurrences of each unique word in a text file (e.g. *Poker* by Zora Neale Hurston).

- The user types a word and we report *how many times* that word appeared in the book (repeat in a loop until quit).

What would be a good design for this problem?

A. `Map<int, string> wordCounts;`

B. `Map<string, Vector<string>> wordCounts;`

[✓ C. `Map<string, int>  wordCounts;`]

D. `Map<string, Vector<int>> wordCounts;`

E. Other/none/more

Write a program to count the number of occurrences of each unique word in a text file (e.g. *Poker* by Zora Neale Hurston).

Map< string, Vector<string>>

How can we record the count?

A. `wordCounts[word] += word;` ✓

[✓ B. `wordCounts[word] += 1;`

C. `wordCounts[word]++;`] ☆

D. B and C are good, but you need to first detect new (never seen before) words so you can start at zero before you start adding +1

E. Other/none/more

↑ not necessary

```
Map<string,int> wordCounts;
string word;
infile >> word;
while (!infile.fail()) {
    //record count here
    infile >> word;
}
```

Hello there.

Write a program to count the number of occurrences of each unique word in a text file (e.g. *Poker* by Zora Neale Hurston).

- Report all words that appeared in the book at least 10 times, in alphabetical order

```
cout << "Most common words:" << endl;
for (string word : wordCounts){
   if (wordCounts[word] >= 10){
      cout << word << "\t";
      cout << wordCounts[word] << endl;
   }
}
```

New (C++11) useful tool!
for loop that iterates over all elements of a container class

Does this work for our alphabetical use case?

- Yes!
- Stanford library Map returns its keys in sorted order