Programming Abstractions

CS106B

Cynthia Lee
Today’s Topics

ADTs

- Map
  - Code example: counting words in text
- Containers-within-containers
  - Shallow copy vs. deep copy
Maps

What are they?
Associative containers

- **Map**
- **Set**
- **Lexicon**

Not as concerned with order but with matching

- **Set**: associates **keys** with **membership** (yes or no)
  - Lexicon: a set of strings, with special optimizations for that
- **Map**: associates **keys** with **values** (could be any type)
Stanford library Map *(selected member functions)*

```cpp
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 the number of occurrences 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<Vector<string>, Vector<int>> 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).

How can we record the count? *(In other words, what goes in the marked place in the code at right?)*

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

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

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

**Does this work for our alphabetical use case?**
- Yes!
- Stanford library Map returns its keys in sorted order
Compound Containers

Containers in containers? It’s turtles all the way down...
Comparing two similar codes:

```cpp
Vector<int> numbers;
numbers.add(1);
numbers.add(2);
numbers.add(3);
Map<string, Vector<int>> mymap;
mymap["abc"] = numbers;
// now we want to add 4, and we have two options
mymap["abc"].add(4);
cout << "New size: " << mymap["abc"].size() << endl;
```

Code option #1
Comparing two similar codes:

```cpp
Vector<int> numbers;
numbers.add(1);
numbers.add(2);
numbers.add(3);
Map<string, Vector<int>> mymap;
mymap["abc"] = numbers;
// now we want to add 4, and we have two options

// Code option #2
Vector<int> test = mymap["abc"];
test.add(4);

cout << "New size: " << mymap["abc"].size() << endl;
```
Comparing two similar codes:

```cpp
Vector<int> numbers;
numbers.add(1);
numbers.add(2);
numbers.add(3);
Map<string, Vector<int>> mymap;
mymap["abc"] = numbers;
// now we want to add 4, and we have two options
mymap["abc"].add(4);

Code option #1

cout << "New size: " << mymap["abc"].size() << endl;

Code option #2

Vector<int> test = mymap["abc"];
test.add(4);
```

Predict the outcome of each option (what is printed?):
(A) Both print 3  (B) Both print 4  (C) One prints 3, other prints 4  
(D) Something else or error
Comparing two similar codes:

You don’t need to worry too much about the details of how the two cases differ in terms of behind-the-scenes mechanism—I just wanted to flag it as a potential issue in case you accidentally encounter this in your code.

```cpp
class Vector<
    int> numbers;
numbers.add(1);
numbers.add(2);
numbers.add(3);

Map<
    string, Vector<
    int>> mymap;
mymap["abc"] = numbers;

// now we want to add 4, and we have two options
mymap["abc"].add(4);
```

Code option #1

```cpp
Vector<
    int> test = mymap["abc"];
test.add(4);
```

Code option #2

```
cout << "New size: " << mymap["abc"].size() << endl;
```

Predict the outcome of each option (what is printed?):

(A) Both print 3  (B) Both print 4  (C) One prints 3, other prints 4  (D) Something else or error