Programming Abstractions

CS106B

Cynthia Bailey Lee
Julie Zelenski
Today’s Topics

More ADTs!

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

(not like the driving directions kind of maps though)
Associative containers

- Map
- Set
- Lexicon

Not as concerned with order but with association

- Map: associates **keys** with **values** (each could be any type)
- Set: associates **keys** with **membership** (in or out)
  - Lexicon: a set of strings, with special internal optimizations for that

Stanford University
Stanford library Map *(selected member functions)*

```cpp
void put(KeyType& key, ValueType& value);
bool containsKey(KeyType& key);
ValueType get(KeyType& key);
ValueType operator [] (KeyType& key);
```

```cpp
#include "map.h"

Map<string, string> phone; // Map takes two(!) template parameters

phone["Cynthia"] = "321-101-0000"; // two syntax options for adding new item
phone.put("Julie", "878-333-1234");

if (phone.containsKey("Cynthia") && phone.containsKey("Julie")) {
    cout << phone["Cynthia"] << endl; // two syntax options for getting item
    cout << phone.get("Julie") << endl;
    cout << phone["MTL" ] << endl; // what would this do??
}
```
Map Code Example

Tabulating word counts
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). Then do two things with those counts:

- **Words that occur many times:**
  - Print all words that appeared in the book at least 100 times, in alphabetical order

- **Query mode:**
  - Give us a word and we report *how many times* that word appeared in the book (potentially call this function to query the counts many times)
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 gives us a word and we report how many times that word appeared in the book (potentially many such queries).

What would be a good design for this problem?

A. `Map<int, string> wordCounts;`
B. `Map<Vector<string>, Vector<int>> wordCounts;`
C. `Map<Vector<int>, Vector<string>> wordCounts;`
D. `Map<string, int> wordCounts;`
E. `Map<string, Vector<int>> wordCounts;`
F. 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 place marked “record count here” 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
// We are given a vector that is just the book, broken into pieces based on spaces between words. The type is:
// Vector<string> words;

Map<string, int> wordCounts;
for (string word : words) {
    // record count here
}
```
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).

```cpp
// userWord is a word the user typed into the console
cout << userWord << " appears " << wordCounts[userWord] << " times" << endl;
```

**What happens if queryWord is not a word in the book?**

- Will the program crash?
- What other issue(s) besides crash do you foresee?
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 100 times, in alphabetical order

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

**Does this work for our alphabetical order requirement?**
- Yes!
- Stanford library Map returns its keys in sorted order
How do we measure “faster” in Computer Science?

NOT AS SIMPLE AS YOU MIGHT THINK...
Recall our discussion of performance with the Vector add vs. Insert...
Your turn: Vector performance

- Answer: (D) Something else! (about 50x)
  - In addition to analyzing the code and predicting number of writes needed, we can also time the code using our Stanford 106B test system.
  - Check the code bundle for class today for runnable version!

```cpp
/* * * * * * Test Cases * * * * * */
PROVIDED_TEST("Timing comparison")
{
    int size = 500000;
    TIME_OPERATION(size, runInsert(size));
    TIME_OPERATION(size, runAdd(size));
}

void runInsert(int size)
{
    Vector<int> v;
    for (int i = 0; i < size; i++) {
        v.insert(0, i);
    }
}

void runAdd(int size)
{
    Vector<int> v;
    for (int i = 0; i < size; i++) {
        v.add(i);
    }
}
```

Performance analysis technique 1:
Benchmarking (actually run it, and time it)
Your turn: Vector performance

**Answer: (D) Something else! (about 50x)**

- Number of times a number is written in a box:
  - **OPTION 1:**
    - First loop iteration: 1 write
    - Next loop iteration: 2 writes … continued…
    - Formula for sum of numbers 1 to N = (N * (N + 1)) / 2
    - *(don’t worry if you don’t know this formula, we only expected a ballpark estimate)*
    - 100 * (100 + 1 ) / 2 = 10,100 / 2 = **5,050**
  - **OPTION 2:**
    - First loop iteration: 1 write
    - Next loop iteration: 1 write … continued…
    - **100**

Performance analysis technique 2: Counting the precise number of writes to memory
Big-O: our primary performance analysis technique

- Big-O analysis in computer science is a way of counting the number of “steps” needed to complete a task
  - Doesn’t really consider how “big” each step is
  - Doesn’t consider how fast the computer’s CPU or other hardware components are
  - Doesn’t involve any actual measurement of the time elapsed for any real code in any way
- But despite all that, really useful for making broad comparisons between different approaches
Efficiency as a virtue?

- In computer science, we tend to obsess about efficiency, but it’s worth taking a step back and asking ourselves, is efficiency always a virtue?
  - Racing to be first to the finish line, but with an answer that’s wrong, isn’t helpful!
  - That might seem obvious, but it happens *all the time* in real tech products.
Another example...
The danger of a cheap solution: Twitter cropping

In the summer of 2020, Twitter users noticed something strange about Twitter’s new photo cropping algorithm that is supposed to choose the most important/interesting part of the image.

Given a too-tall image, it selects which part to show.

It picked the Senator McConnell (the white man), not President Obama.
The danger of a cheap solution: Twitter cropping

In the summer of 2020, Twitter users noticed something strange about Twitter’s new photo cropping algorithm that is supposed to choose the most important/interesting part of the image.

Given a too-tall image, it selects which part to show.

It picked the Senator McConnell (the white man), not President Obama.
Efficiency as a virtue?

- In each of these cases, companies chose an algorithm that would reach the desired product outcome in a way that is the most efficient, for some business and/or computer science definition of efficient, but came up with answers that were “wrong” (problematic) in ways that are significant for society.
- How can we balance cost (which is what efficiency is really about in capitalism) with correctness and justice for society?
- Reflect on this in your Assignment 2!