Programming Abstractions

CS106B

Ashley Taylor

Thanks to Cynthia Lee and Marty Stepp for letting me adapt their slides

Today's Topics

Finishing our discussion of Stacks

Queues

- The "opposite" of a Stack

File Input and Output

- A new way to handle data
- countUniqueWords example

Next Time:

- Sets and Lexicons
- Maps

Stacks

A Stack is an ADT best at accessing elements in the **reverse** order from how they were added.

Review from Monday: Stacks

- "Last In, First Out"
  - The most recently added item is the most accessible
- Operations
  - `peek()`
  - `push(elem)`
  - `pop()`
- Comparison to Vector
  - Worse if you need to access arbitrary elements
  - Better if you only need the last thing added (or if you need to "reverse" your ADT)

https://c2.staticflickr.com/8/7583/15638298618_104af94267_b.jpg

Review: Reverse Polish Notation

Alternative way of writing arithmetic operations where the operator is after the arguments

Postfix (RPN):

- 4 3 * 4 3 * +

Equivalent Infix:

- (4*3) + (4*3)

http://commons.wikimedia.org/wiki/File:Hewlett-Packard_48GX_Scientific_Graphing_Calculator.jpg

Reverse Polish Notation

This RPN expression:

- 4 3 * 7 2 5 * + +

Is equivalent to which infix expression?

A. ((4*3) + (7*2)) + 5
B. (4*3) + ((7+2) + 5)
C. (4*3) + (7 + (2*5))
D. Other/none/more than one

http://commons.wikimedia.org/wiki/File:Hewlett-Packard_48GX_Scientific_Graphing_Calculator.jpg

Reverse Polish Notation

How can we use a Stack to evaluate RPN expressions?

Postfix (RPN):

- 4 3 * 4 3 * +

Equivalent Infix:

- (4*3) + (4*3)

http://commons.wikimedia.org/wiki/File:Hewlett-Packard_48GX_Scientific_Graphing_Calculator.jpg

Queues

A Queue is an ADT best at accessing elements in the **same** order as they were added.

Queues

- "First In, First Out"
  - The oldest item is the most accessible
- Useful whenever you care about maintaining order
- Operations
  - `peek()`
  - `enqueue(`***`elem`***`)`
  - `dequeue()`

https://c1.staticflickr.com/5/4073/4823442859_b954c3ec29_b.jpg

Queue Example (CodeMystery3 on CodeStepByStep)

- What is the output of the following function when we give input {42, -3, 4, 15, 9, 71}

```
void collectionMystery(Queue<int>& q) {
    Stack<int> s;
    int size = q.size();
    for (int i = 0; i < size; i++) {
        int n = q.dequeue();
        if (n % 2 == 0 {
            s.push(n);
        } else {
            q.enqueue(n);
        }
    }
    cout << "q=" << q << endl;
    cout << "s=" << s << endl;
}
```

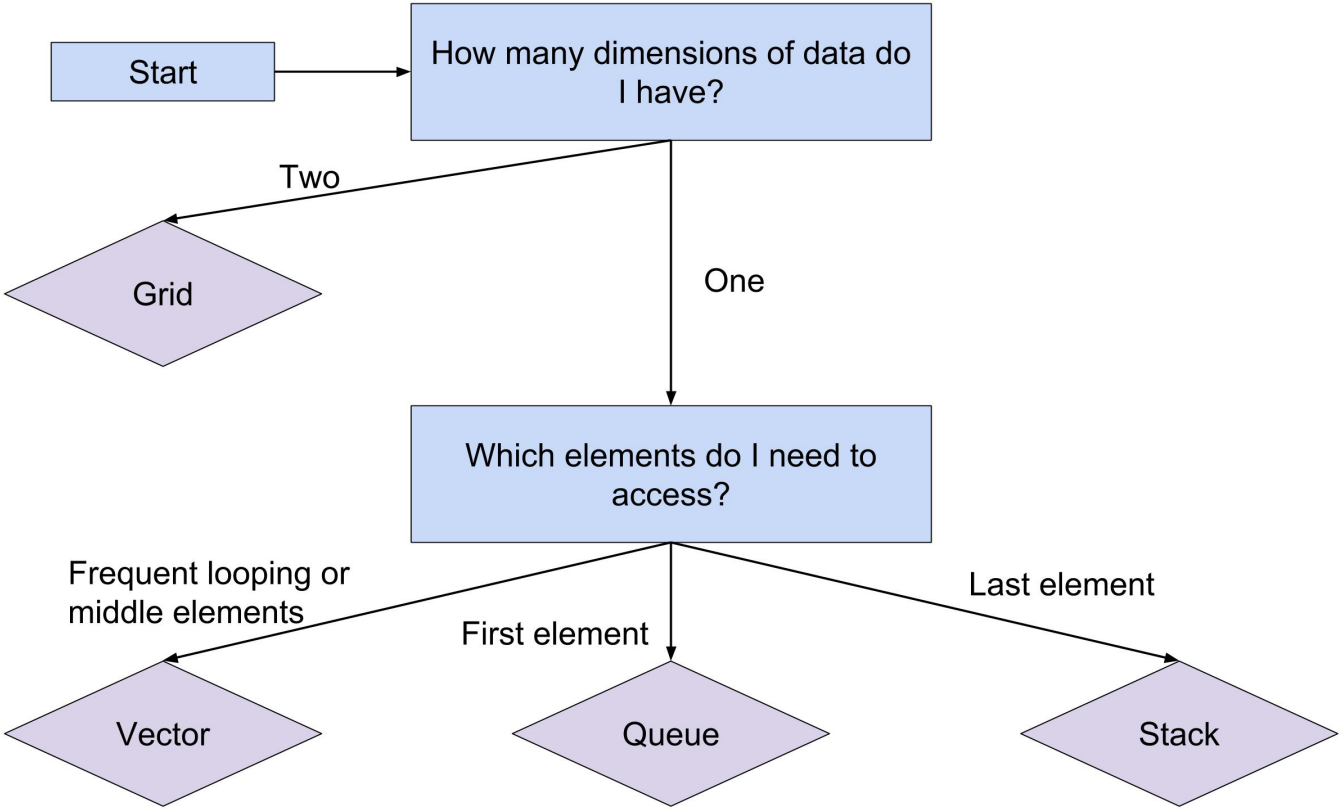
Queue Example (CodeMystery3 on CodeStepByStep)

- What is the output of the following function when we give input {42, -3, 4, 15, 9, 71}

```
void collectionMystery(Queue<int>& q) {
    Stack<int> s;
    int size = q.size();
    for (int i = 0; i < size; i++) {
        int n = q.dequeue();
        if (n % 2 == 0 {
            s.push(n);
        } else {
            q.enqueue(n);
        }
    }
    cout << "q=" << q << endl;
    cout << "s=" << s << endl;
}
```

q={-3, 15, 9, 71}
s={42, 4}

ADT Soup


Start
How many dimensions of data do I have?
Two
Grid
One
Which elements do I need to access?
Frequent looping or middle elements
First element
Last element
Vector
Queue
Stack

Announcements

Sections start this week!

- Go to cs198.stanford.edu to see your section time and room
- The section handout is on the website. You can also view all the problems at https://codestepbystep.com/problemset/view?id=3

Assignment 1 YEAH Hours!

- Lots of tips and tricks for the first assignment – go check it out at https://mvideox.stanford.edu/Course/969
- Special thanks to SL Garrick
- Stay tuned for more info about Assignment 2 YEAH hours (which will be early next week)

Midterm and Final Exam Conflicts

- Please fill out this form (https://goo.gl/forms/uUdxpaGki8yrsr9i1) by the end of the week, regardless of whether you have a conflict or not.

Honor Code – Summary

- Don’t look at anyone else’s code (for any class, even CS106A)
- Don’t show your code to anyone else (except CS106 Section Leaders, Cynthia, or me)
- Cite any help you get

File Input and Output

File input is a new (and easier) way of getting data into your program, compared to user input. File output lets you save data beyond the program (before, we were just printing)

Files

- Store data beyond the run of a program
- Easy way to gather a lot of information together (vs. user input)
- Stored in **streams** in C++
  - Similar to strings – sequence of characters
  - To read files, declare an `ifstream` (**i**nput **f**ile **stream**)
    - Similar to `cin`
  - To write to files, declare an `ofstream` (**o**utput **f**ile **stream**)
    - Similar to `cout`

Common File Reading/Writing Pattern

- Open File
  - `#include <fstream>`
  - `#include "filelib.h"`
    - `string promptUserForFile(`***`stream`***`,`
      `prompt``)`
  - If you already have the filename:
    - ***`stream`***`.open("`***`file.txt`***`")`
- Read/write to file (more on that soon)
- Close the file
  - `stream.close()`

Reading from files – character by character

```
ifstream infile;
promptUserForFile(infile, "File?");

char ch;

while(infile.get(ch)) {
     // do something with ch;
}

infile.close();
```

Reading from files – line by line

```
ifstream infile;
promptUserForFile(infile, "File?");

string line;

while(getline(infile, line)) {
    // do something with line
}

infile.close();
```

- `getline` should be all lowercase

Reading from files – formatted input

```
ifstream infile;
promptUserForFile(infile, "File?");

string word;

while(infile >> word) {
    // do something with word
}

infile.close();
```

- Warning: Don't try to mix line by line input with formatted input
- Works with other types too
- Automatically strips whitespace

Writing to files

```
ofstream outfile;
promptUserForFile(outfile, "File?");

// an example of character output
outfile.put('a');

// most of your output would be like this
int num = 1;
outfile << num << " example of formatted output" << endl;

outfile.close();
```

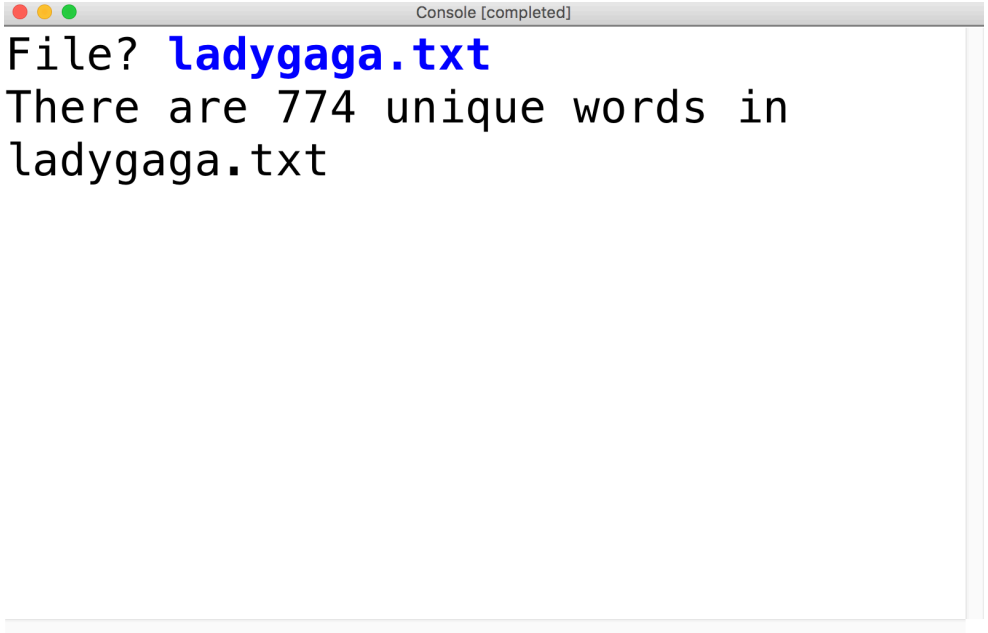
Common File I/O Gotchas

- What's wrong with this code?

```
ifstream infile;
promptUserForFile(infile, "File?");

string line;

while (!infile.fail()) // or !infile.eof()
{
    getline(infile, line);
    // do something with line
}

infile.close();
```

File Reading Example: CountUniqueWords

- One basic statistic about a text is the number of unique words it has
  - Linguists and computer scientists frequently start analysis with the number of unique words
  - Good indication of vocabulary
- Problem: how can we determine the number of unique words in a file?



CountUniqueWords: Which ADT?