Programming Abstractions

CS106X

Cynthia Lee

Today's Topics

ADTs

- Stack
  - Example: Reverse-Polish Notation calculator
- Queue
  - Example: Mouse Events

Stacks

New ADT: Stack

```
template <typename ValueType> class Stack {
public:
   Stack();
   virtual ~Stack();
   int size() const;
   bool isEmpty() const;
   void clear();
   void push(ValueType value);
   ValueType pop();
   ValueType peek() const;
   std::string toString();
private:
   -Redacted-

};
```

Source: http://www.flickr.com/photos/35237093334@N01/409465578/
Author: http://www.flickr.com/people/35237093334@N01 Peter Kazanjy]

Using a Stack to buffer file input

```
void mystery(ifstream& infile) {
    Stack<string> lines;
    string line;
    while (getline(infile,line)) {
        lines.push(line);
    }
    infile.close();
    while (!lines.isEmpty()) {
        cout << lines.pop()
             << endl;
    }
}
```

What does this code do?

A. Prints a file's contents to cout
B. Prints only the first line of a file to cout
C. Prints out a file to cout in reverse
D. All/ none/ more

Using a Stack to buffer file input

```
void mystery(ifstream& infile) {
    Stack<string> lines;
    string line;
    while (getline(infile,line)) {
        li
    }
    infile
    while
        cout << lines.pop()
             << endl;
    }
}
```

Why do I need Stack?
I could have done that with a Vector!

What does this code do?

A. Prints a file's contents to cout
B. Prints only the first line of a file to cout

ut a file to cout in

e/ more

Stack or Vector?

```
void mystery(ifstream& infile) {
    Stack<string> lines;
    string line;
    while (getline(infile,line)) {
        lines.push(line);
    }
    infile.close();
    while (!lines.isEmpty()) {
        cout << lines.pop()
             << endl;
    }
}
```

```
Vector<string> lines;
```

```
lines.insert(lines.size(), line);
```

```
cout << lines[lines.size()-1]
     << endl;
lines.remove(lines.size()-1);
```

Vector version

```
void mystery(ifstream& infile) {
    Vector<string> lines;
    string line;
    while (getline(infile,line)) {
        lines.insert(lines.size(),line);
    }
    infile.close();
    while (!lines.isEmpty()) {
        cout << lines[lines.size()-1]
             << endl;
        lines.remove(lines.size()-1);
    }
}
```

This code isn't terrible, but it is harder to read quickly, and is probably more error prone.

For example, it would be easy to forget the "-1" when you remove "lines.size()-1".

Applications of Stacks

We've seen one (buffering input and giving it back in reverse—LIFO—order). What else are Stacks good for?

Operator Precedence and Syntax Trees

Ignoring operator precedence rules, how many distinct results are there to the following arithmetic expression?

- 3 * 3 + 3 * 3

Reverse Polish Notation

Ambiguities don't exist in RPN

Also called "postfix" because the operator goes after the operands

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 postfix expression:

- 4 3 * 7 2 5 * + +

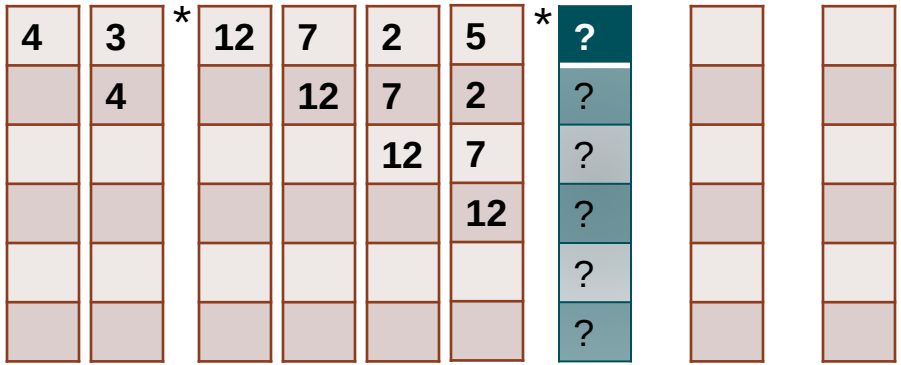
Is equivalent to this infix expression:

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

Stacks and RPN

- Evaluate this expression with the help of a stack
  - Encounter a **number**: **PUSH** it
  - Encounter an **operator**: **POP** two numbers and **PUSH** result
- 4 3 * 7 2 5 * + +



Contents of the stack, reading from top down:

(A) 7, 12

(C) 10, 7,12

(D) 10, 5, 2, 7, 12

(E) Other

Stacks and RPN: What does that look like in code?

Evaluate this expression with the help of a stack

- Encounter a **number**: **PUSH** it
- Encounter an **operator**: **POP** two numbers and **PUSH** result

43*725*++

```
/* Note: this code assumes numbers are all 1 digit long */
bool evaluate(Stack<int>& memory, string instruction) {
    for (int i = 0; i < instruction.size(); i++) {
        if (isdigit(instruction[i])) {
            int value = instruction[i] - '0'; //convert char->int
            memory.push(value);
        } else if (isSupportedOperator(instruction[i])) {
            if (memory.size() < 2) return false;
            int second = memory.pop();
            int first = memory.pop();
            int result = compute(first, instruction[i], second);
            memory.push(result);
        } else {
            return false;
        }
    }
    return memory.size() == 1; //validity check
}
```

Application of Stacks

WE'VE SEEN ONE (BUFFERING INPUT AND GIVING IT BACK IN REVERSE—LIFO—ORDER). WHAT ELSE ARE STACKS GOOD FOR?

Operator Precedence

Ignoring operator precedence rules, how many distinct results are there to the following arithmetic expression?

- 3 * 3 + 3 * 3

- A pain to have to learn those rules
- Or use parentheses to disambiguate

Reverse Polish Notation

Ambiguities don't exist in RPN! ☺

Also called "postfix notation" because the operator goes after the operands

Postfix (RPN):

- 4 3 * 3 +

Equivalent Infix:

- (4*3) + (3)

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

Reverse Polish Notation

This postfix expression:

- 4 3 * 7 2 5 * + +

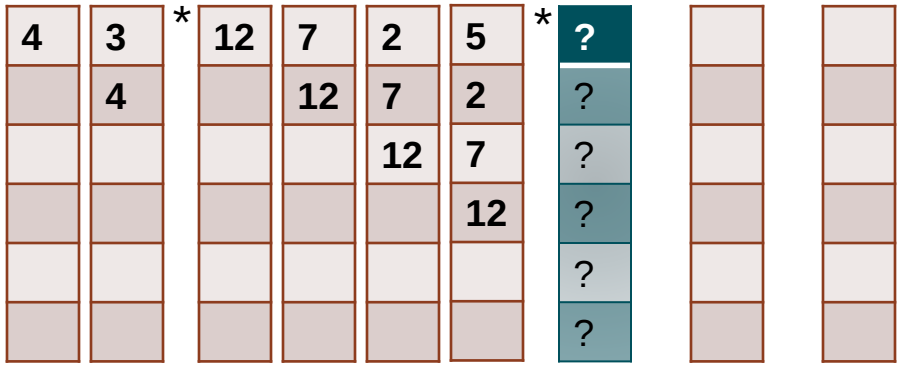
Is equivalent to this infix expression:

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

Stacks and RPN

- Evaluate this expression with the help of a stack
  - Encounter a **number**: **PUSH** it
  - Encounter an **operator**: **POP** two numbers and **PUSH** result
- 4 3 * 7 2 5 * + +



Contents of the stack, reading from top down:

(A) 7, 12

(B) 10, 7,12

(C) 10, 5, 2, 7, 12

(D) Other

Stacks and RPN: What does that look like in code?

Evaluate this expression with the help of a stack

- Encounter a **number**: **PUSH** it
- Encounter an **operator**: **POP** two numbers and **PUSH** result

43*725*++

```
/* Note: this code assumes numbers are all 1 digit long */
bool evaluate(Stack<int>& memory, string instruction) {
    for (int i = 0; i < instruction.size(); i++) {
        if (isdigit(instruction[i])) {
            int value = instruction[i] - '0'; //convert char->int
            memory.push(value);
        } else if (isSupportedOperator(instruction[i])) {
            if (memory.size() < 2) return false;
            int second = memory.pop();
            int first = memory.pop();
            int result = compute(first, instruction[i], second);
            memory.push(result);
        } else {
            return false;
        }
    }
    return memory.size() == 1; //validity check
}
```

Queues

WHAT ARE THEY?
EXAMPLE APPLICATION

Queues

They work the same way a waiting in line (or, if you're in the UK, *queuing*) works.

FIFO = "First in, first out"

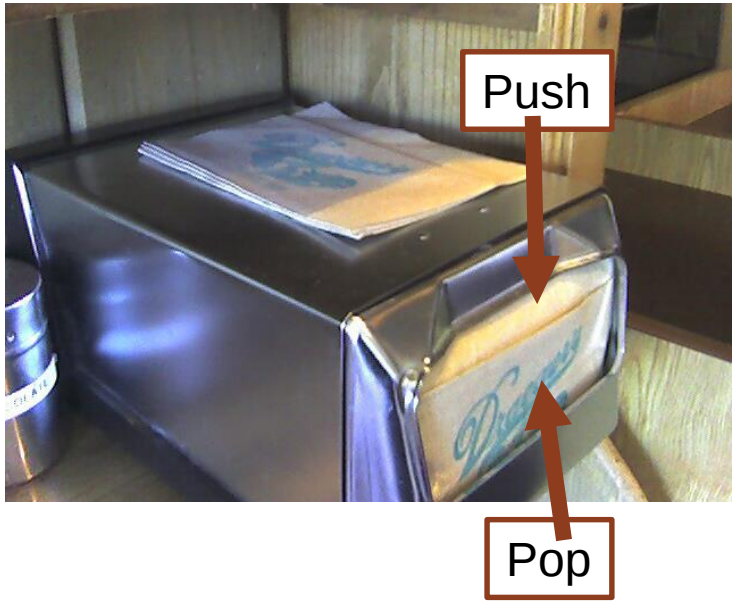
"First come, first serve"

Waiting for Apple Watch

Where Stack and Queue are accessed

This may seem obvious, but it's an important point for the behind-the-scenes code implementation of these data structures:

Stack: only accessed on one end



Queue: accessed at both ends

Event queues (used in Fauxtoshop)

While your code executes, a separate part of the program is constantly running, and its only job is listening for events and recording them

- Mouse moved, mouse clicked, mouse dragged, etc

Every once in a while, your code can call **`getNextEvent()`** to see what has happened

- `getNextEvent()` returns the events one at a time, in the same order they happened
  - In other words, returns them in **FIFO** order
  - When it is "recording" events, it is **enqueuing** events in an event **QUEUE**

Very common use of the Queue ADT

Map

WHAT ARE THEY?
EXAMPLE APPLICATION

Linear containers, AKA **Sequential** containers

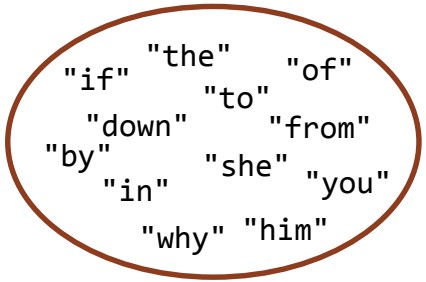
- Array
- Vector
- Stack
- Queue

The question "Which is the [1st/2nd/3rd…] element?" is salient

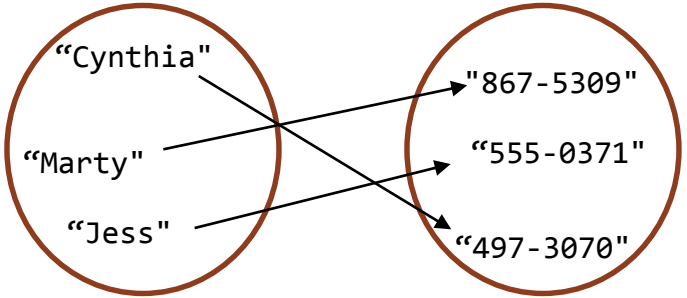
- *Note:* even though we can't directly access 3rd element with Stack/Queue, there is a 3rd element—there is an ordering

Associative containers

- Map
- Set
- Lexicon



set



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)

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).

- Report all words that appeared in the book at least 10 times, in alphabetical order
- Allow the user to type a word and report *how many times* that word appeared in the book

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).

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

```
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

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

Does this code do the job?

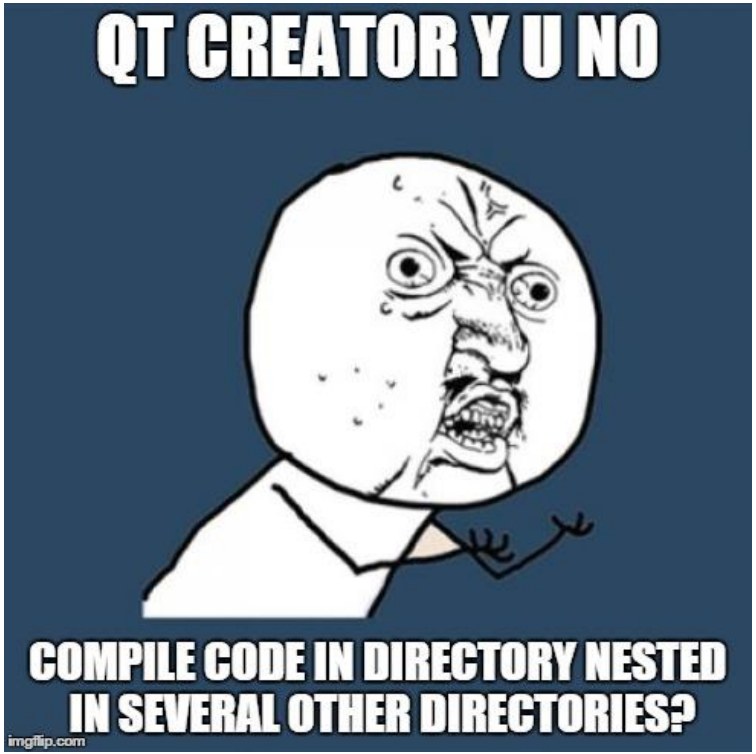
A. YES
B. NO
C. YES except for the detail of printing in alphabetical order
D. Other/none/more

QT Creator

A FEW WARNINGS & TIPS

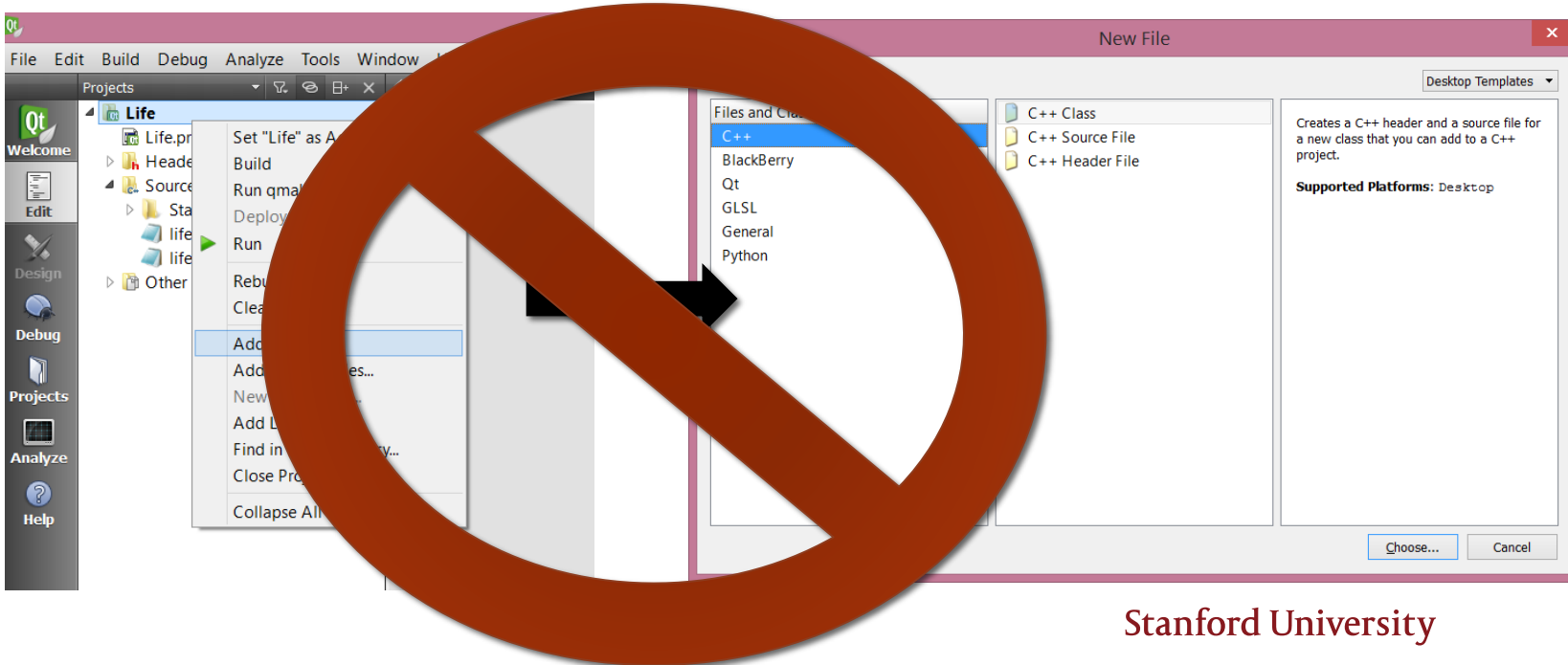
If your code doesn't compile and gives an error about unknown function Main() (note capital M):

- There is more than one reason this could arise, but the most common by far is that QT lost its ability to find the path to our Stanford library includes (which define a fake Main()), because you put your code in a directory/folder that is deeply nested on your drive.
- For example, C:\Documents and Settings\Documents\classes\Stanford\2015\Summer\CS106B\assignments\C++\Assignment1\...
- **You need to move the assignment directory closer to C:\ (or on mac, the root directory)**



If your code doesn't compile and gives you "multiple definition" errors for all the functions you have:

- This can arise if you add new files to your code inside QT creator using its handy "Add New…" feature.
- NOT HANDY! Do not use this feature. It breaks the .pro file.



If it's too late and you already used the "Add New" feature

1. QUIT: Close QT Creator
2. CLEAN UP BROKEN FILES: Delete the .pro and .pro.user files in your code directory
3. GRAB REPLACEMENT: Download/unzip the assignment bundle again, and get a fresh copy of the project file (the one that ends in .pro) and put it in place of the one you deleted.
4. ALL BETTER! Re-open QT Creator (it will make a new .pro.user)