CS 106B, Lecture 4
File I/O and Debugging

Based on slides created by Marty Stepp, Chris Gregg, Keith Schwarz, Julie Zelenski, Jerry Cain, Eric Roberts, Mehran Sahami, Stuart Reges, Cynthia Lee, and others

Plan for Today

- Learn about another form of input and output: files
- Debugging strategies

File I/O

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**)
  - To write to files, declare an `ofstream` (**o**utput **f**ile **stream**)
    - Similar to cout

Common File I/O Pattern

- Open File
  - #include <fstream> // standard library pkg for files
  - #include "filelib.h" // contains helpful methods
    - string promptUserForFile(***stream, prompt)***

      // asks user for filename and opens the file in ***stream***
  - If you already have the filename:
    - ***stream***.open("***file.txt***")
- Read/write to file (more on that soon)
- Close the file
  - stream.close()

Creating and Closing

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

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

infile.close();
```

Same for every file-reading program
Creates `ifstream` object
Closes `ifstream` object

Opening File

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

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

infile.close();
```

Asks for the user for the filename

Opening File Alternative

```
ifstream infile;
infile.open("File.txt");

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

infile.close();
```

Good when **you** know the file to open

Reading Char by Char

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

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

```
infile.close();
```

Declare the variable to read data into (ch)
While loop continues **until read fails**
- Every iteration of while loop is new char

Reading Line by Line

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

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

```
infile.close();
```

Now reads each **line** (breaks on newline characters)
Still declare the line before the while loop
Still continues until getline fails; each while loop iteration has a different line
Notice lowercase l of getline

Reading Formatted Input

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

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

```
infile.close();
```

Now reads each **word** (removes whitespace)
Still declare the wordbefore the while loop
Still continues until fails to read a new word
each while loop iteration has a different word
Works with other types (Vector or int, e.g.) too
Don't try to mix with getline

Writing Output

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

string word = "output";
int x = 3;
outfile << word << x;

outfile.close();
```

Similar to reading formatted input
Works a lot like cout
use <<
Works with (basically) any type

Use `ofstream` instead `ifstream`

Announcements

- Assignment 0 due **tomorrow at 5PM**
- Assignment 1 (Game of Life) will be released today; due **Thursday, July 5, at 5PM**. You can work in a pair.
  - **Honor Code Reminder**: Please review the Honor Code handout on the course website before beginning this assignment
  - Any student who is found in violation of the Honor Code will fail the course in addition to sanctions applied by OCS
- No class on July 4th – if you have section, either attend a Thursday or Friday section or watch the videoed section and email your SL a summary

Debugging

Steps to Debugging

- Determine that you have a bug
- Isolate the bug's location
- Find the culprit code

Identifying a bug

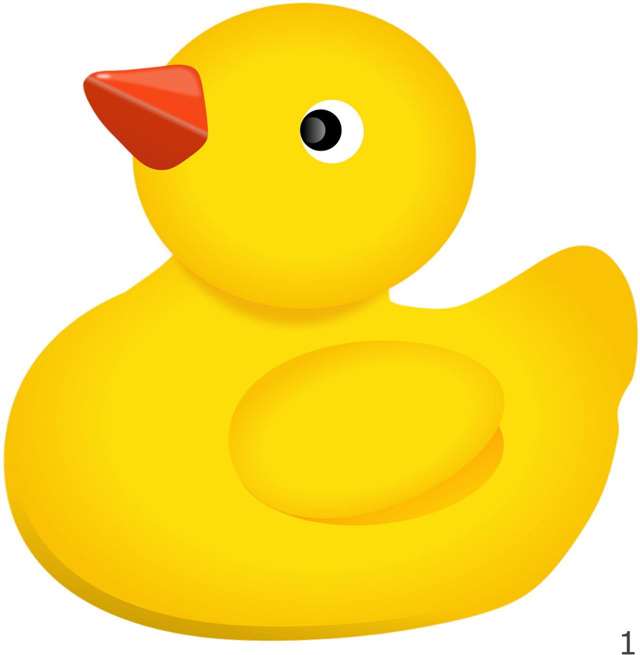
- In order to find a bug, lots and lots of testing (more on that on Tuesday)
- What is the behavior that you think is buggy (in words)?
- Why do you think that that behavior is buggy?
  - Differs from given expected output?
  - Not what you were expecting?
- Under what circumstances does the bug appear?
  - Try different inputs or outputs
  - Goal: find the smallest output possible that reproduces the bug
- Be specific!

Isolating the Bug

- Goal: where in the code could the bug be?
- Be creative – better to think of too many places than too few
- Identify different functions that could be the culprit
  - Then run each function separately
  - Print out parameters and return values
  - Use the debugger!

Finding the Bug

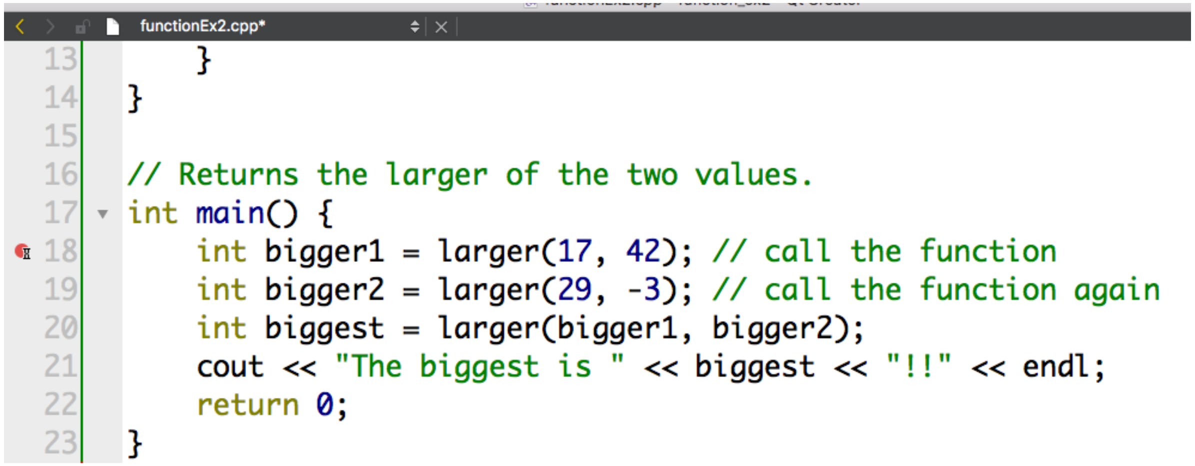
- Once you've found the function, need to find the bug
- What does each line of code do?
  - Use print statements or the debugger to verify your assumptions
  - Explain each line of code to your partner or an inanimate object
- Draw pictures – keep track of values in data structures and variable values
- If you still can't find it, get help!
  - LaIR
  - OH



Source: https://pixabay.com/en/duckling-toys-yellow-rubber-duck-2542277/

Using the Debugger

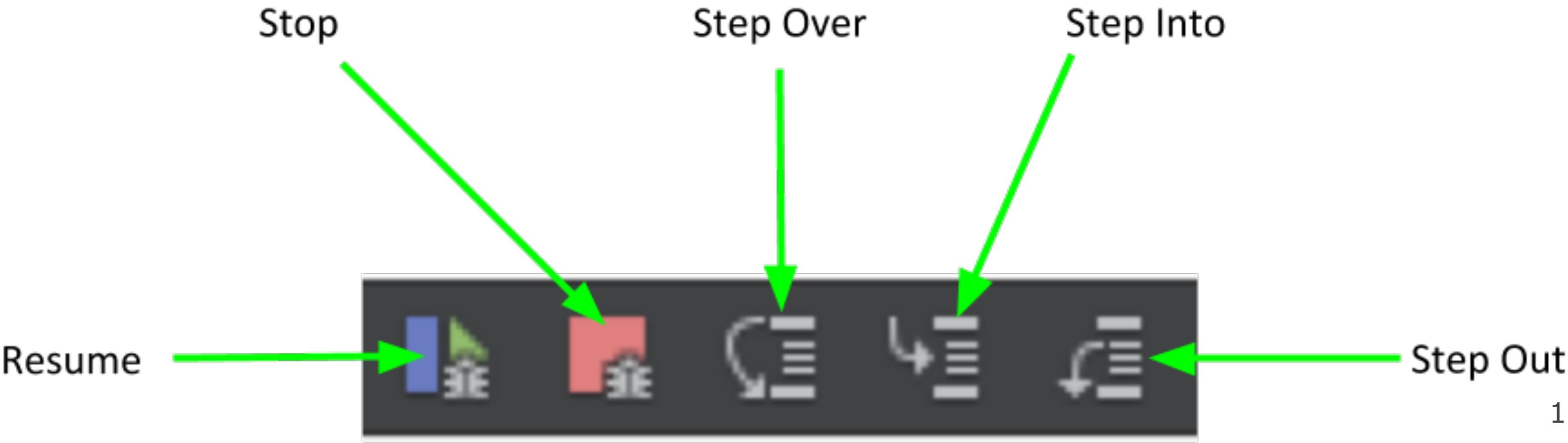
- Add a breakpoint – program will pause at that line of code



```
    }
}

// Returns the larger of the two values.
int main() {
    int bigger1 = larger(17, 42); // call the function
    int bigger2 = larger(29, -3); // call the function again
    int biggest = larger(bigger1, bigger2);
    cout << "The biggest is " << biggest << "!!" << endl;
    return 0;
}
```

- "Step" through code execution, line by line



Print Debugging

- Alternative to debugger – personal choice (debugger is more powerful, but doesn't represent collections well)
- Idea: print relevant information at every line
- Tips for good print debugging
  - Give good messages at each line (slightly longer, but WAY better output)
  - Print variable values WITH the variable name
  - Debug a section at a time (can be overwhelming otherwise)
  - Add if statements to conditionally print

Debugging Example

- Please don't say the bug – this exercise is for good debugging practices
- What are some smaller inputs we could try?
- Which variables should we track?
- Which lines should we examine?