Outline

- This is the fifth in a sequence of six topics on
  - C assertions
  - Code development strategies
  - Testing
  - Commenting your code
  - Using print statements for debugging
  - Using tracing for debugging

Outline

- In this topic, we will:
  - Describe the purpose of logging
  - Consider how it can be used for debugging
  - Explain its limitations

Purpose of logging

- You've written a function, it compiles, and…
  …it doesn't work
  - The easiest step is to log your code with `std::cout`

Logging with high-low

- Here is a version of the high-low game where the computer plays:

```
void high_low() {
    int lo{1};
    int hi{100};

    while ( true ) {
        int guess{ lo/2 + hi/2 };

        std::cout << "My guess is " << guess << std::endl;
        std::cout << "Is my guess low (l), correct (c) or high (h)? ";
        char correctness{};
        std::cin >> correctness;
```


Logging with high-low

```
        if ( (correctness == 'c') || (correctness == 'C') ) {
            return;
        } else if ( (correctness == 'l') || (correctness == 'L') ) {
            lo = guess + 1;
        } else if ( (correctness == 'h') || (correctness == 'H') ) {
            hi = guess - 1;
        } else {
            std::cout << "You entered '" << correctness << "', "
                      << "please try again" << std::endl;
    }

    // We should not get here
    assert( false );
}
```


Logging with high-low

- For example, suppose your number is 35

```
My guess is 50
Is my guess low (l), correct (c) or high (h)? h
My guess is 24
Is my guess low (l), correct (c) or high (h)? l
My guess is 36
Is my guess low (l), correct (c) or high (h)? h
My guess is 29
Is my guess low (l), correct (c) or high (h)? l
My guess is 32
Is my guess low (l), correct (c) or high (h)? l
My guess is 33
Is my guess low (l), correct (c) or high (h)? l
My guess is 34
Is my guess low (l), correct (c) or high (h)? l
My guess is 34
Is my guess low (-1), correct (0) or high (1)?
```


Logging with high-low

- A single `std::cout` statement can help you see what is happening:

```
while ( true ) {
    int guess{ lo/2 + hi/2 };

    std::cout << " >>> " << lo << ", " << hi << ", "
                         << guess << std::endl;

    std::cout << "My guess is " << guess << std::endl;
    // Other code from the while loop body...
}
```

Logging with high-low

- With this logging statement, we see what is happening:

```
 >>> 1, 100, 50
My guess is 50
Is my guess low (l), correct (c) or high (h)? h
 >>> 1, 49, 24
My guess is 24
Is my guess low (l), correct (c) or high (h)? l
 >>> 25, 49, 36
My guess is 36
Is my guess low (l), correct (c) or high (h)? h
 >>> 25, 35, 29
My guess is 29
Is my guess low (l), correct (c) or high (h)? l
 >>> 30, 35, 32
My guess is 32
```


Logging with high-low

```
 >>> 30, 35, 32
My guess is 32
Is my guess low (l), correct (c) or high (h)? l
 >>> 33, 35, 33
My guess is 33
Is my guess low (l), correct (c) or high (h)? l
 >>> 34, 35, 34
My guess is 34
Is my guess low (l), correct (c) or high (h)? l
 >>> 35, 35, 34
My guess is 34
Is my guess low (-1), correct (0) or high (1)?
```


Logging with high-low

- You determine the problem is with the guess:

```
int guess{ lo/2 + hi/2 };
```

- You remember that division truncates,
  so `35/2 + 35/2` evaluates to `17 + 17` which equals `34`

- You try the following, instead:

```
int guess{ (lo + hi)/2 };
```


Purpose of logging

- Using `std::cout` or logging is a simple debugging technique
  - It is fast and dirty
  - However, it will be more difficult to find more complex bugs
    - If you don't know where the issue is,
      you may end up with dozens of print statements
  - You must remember to comment out or delete all such logging statements prior to submitting your code
    - Otherwise, it will likely fail

Purpose of logging

- Logging is used in industry
  - Many applications keep a log file
  - That log file contains data from the execution of the application
  - If the user submits a bug report,
    the log file can be examined to help track down the bug
  - Of course
    - If you don't log the right variables, this will not help
    - If you log too much information,
      this will use up the user's memory

Summary

- Following this lesson, you now:
  - Have an idea of how to debug using `std::cout`
  - Understand this will work to find simple bugs
    - It is even used in industry
  - Understand it may be less optimal to find complex or subtle bugs
    - If you don't what to print, it doesn't help
    - You end up printing everything…

References

[1] Wikipedia:
https://en.wikipedia.org/wiki/Log_file#Event_logs

Acknowledgments

None so far.

Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.



Disclaimer

These slides are provided for the ECE 150 *Fundamentals of Programming* course taught at the University of Waterloo. The material in it reflects the authors' best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.