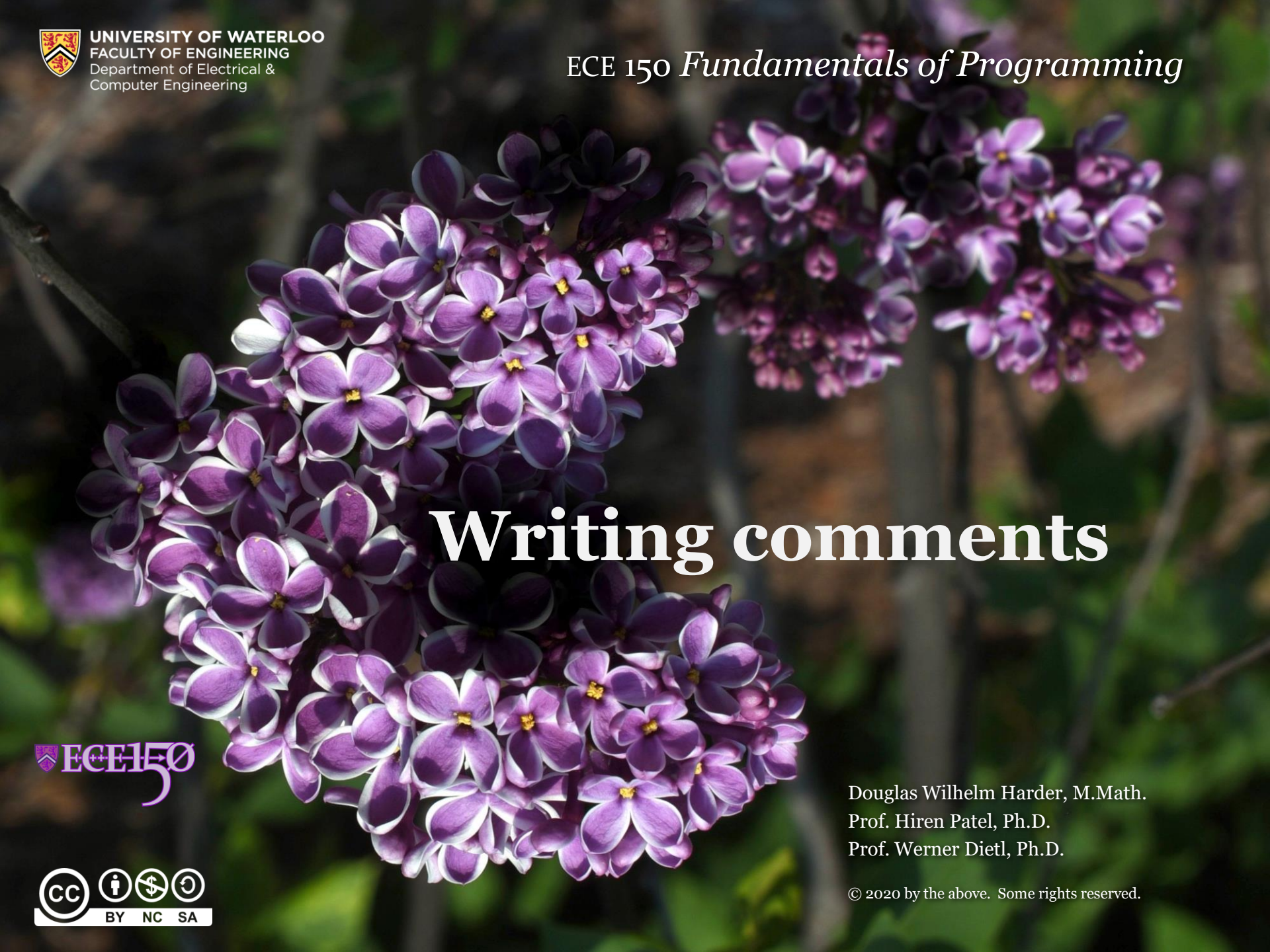
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

ECE 150 *Fundamentals of Programming*



Writing comments

ECE150

Douglas Wilhelm Harder, M.Math.
Prof. Hiren Patel, Ph.D.
Prof. Werner Dietl, Ph.D.

Outline

- This is the third 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 comments
  - Look at how comments can be used for
    - Documentation
    - A description of the functionality
    - A description of the algorithm
    - Explanations of specific oddities in the code

Purpose of comments

- Any code that is useful will be examined and edited many times throughout its existence:
  - Bug fixing
  - Refactoring
  - Performance optimization
  - Feature expansion or enhancements
  - Maintenance
  - Data structure updates
  - Code reuse and modularization
  - Porting
  - Security patching
  - Code review and testing
  - Localization and internationalization
  - Compliance updates
  - Integration

Purpose of comments

- Comments explain to the reader what the code is supposed to accomplish
  - Any good programmer can understand what the code is doing
  - Comments helps the reader understand why the code is doing what it is doing
  - This helps you:
    - Know if the code is actually doing what it should be
    - Allows programmers to discover bugs more easily
    - Allows programmers to extend code

Purpose of comments

- Comments describe the characteristics of a function to the reader
  - These include descriptions that are:
    - Documentary
    - Functional
    - Algorithm
    - Explanatory

- Comments appear either
  - Before the function definition
  - Throughout the function definition

Documentary comments

- Documentation includes:
  - Who was the original author
  - When was the file first written
  - What is the current version number
  - What have been the significant changes made

- Documentary comments generally appear before the function definition

```
// @file      gcd.cpp
// @author    Hiren Patel
// @author    Douglas Wilhelm Harder
// @date      2018-06-19
// @version   1.3
// @since 1.3 Correctly deals with negative arguments
// @since 1.2 Uses 'long' and not 'unsigned long'
// @since 1.1 Fixed bug when one argument is 0
```

Functional comments

- Functionality describe the overall behavior:
  - The types of the parameters and their significance
    - Any restrictions on the arguments
  - What is returned, its type and its relation to the parameters

- Functional comments appear after documentary comments
  - Example:

```
// @param m     the first integer for which the gcd will be calculated
// @param n     the second integer for which the gcd will be calculated
// @returns     the greatest-common divisor (gcd) of the integers m and n
//               - the gcd will always be a positive integer >= 0
```

Algorithmic comments

- Most functions implement some form of algorithm
  - What is the algorithm being used
  - Are there any modifications?
  - Are there any optimizations that implemented here?
  - What steps, if any, are made specifically to deal with C++ types?
  - Additional details, references and comments

- Example:

```
// We will implement the Euclidean algorithm
//   1. If m or n is negative, make them positive--take the absolute value
//   2. If m = n, gcd(m, n) = m, so return m
//   3. If m < n, swap m and n so that m >= n
//   4. Repeat the following:
//     a. Find r such that m = a*n + r
//     b. If r = 0, then gcd( m, n ) = n
//     c. Otherwise, let m take the value n and let n take the value r
// See https://en.wikipedia.org/wiki/Euclidean_algorithm
```

Explanatory comments

- Explanatory comments generally appear in the function definition and describe why something that may be peculiar is done
  - It may emphasize special cases or compiler dependent issues

Comments in the definition

- Algorithmic and explanatory comments appear in the definition

```
int gcd( int m, int n ) {
    // 1. Ensure the parameters are positive
    m = abs( m );
    n = abs( n );

    // 2. Special case: if m = n, return m
    //     - this also deals with gcd( 0, 0 )
    if ( m == n ) {
        return m;
    }

    // 3. If m < n, swap m and n
    if ( m < n ) {
        int tmp{ m };
        m = n;
        n = tmp;
    }
```

Comments in the definition

```
    // 4. Repeat the following:
    while ( true ) {
        // 4a. Find r such that m = a*n + r where a >= 0, r >= 0
        int r{ m%n };

        // 4b. If r = 0, then gcd( m, n ) = n
        if ( r == 0 ) {
            return n;
        }

        // 4c. Otherwise, let m <- n and n <- r
        m = n;
        n = r;
    }

    // We should never get here
    assert( false );
    // Some compilers complain if no return statement appears
    return 0;
}
```

Comments on comments

- Comments should appear before, not after the code in question

```
if ( some-condition ) {
    // The consequent body...
} else {
    // The alternative body...
}

// Do not comment this conditional statement here
```

- Do not align your comments with the code:

```
while ( true ) {
    int r{ m%n };      // 4a. Find r such that m = a*n + r
                       //  where a >= 0, r >= 0
    if ( r == 0 ) {    // 4b. If r = 0,
        return n;      //       then gcd( m, n ) = n
    }

    m = n;             // 4c. Otherwise, let m <- n and n <- r
    n = r;
}
```

Conditional and repetition statements

- Our example doesn't have *interesting* conditional or repetition statements

```
// Summary of the condition, what it tests,
// and why it is testing it
if ( some-condition ) {
    // What to do if the condition is true
} else {
    // What to do if the condition is false
}

// Summary of the for loop, what it is iterating over,
// and why the given range
for ( loop-variable declaration; condition; increment ) {
    // What the loop body is to accomplish
}
```

Poor comments

- Too many students simply describe what the code does in English

```
// Add 2 to n
n += 2;
```

- This is more than useless, as any programmer
  1. Can obviously see this
  2. Now had to waste time reading your comment

- Instead, use something like:

```
// Go on to the next odd integer
n += 2;
```

Summary

- Following this lesson, you now:
  - Have an idea as to how to author comments
  - Understand that there are different types of comments:
    - Documentary
    - Functional
    - Algorithmic
    - Explanatory

References

[1] Wikipedia: https://en.wikipedia.org/wiki/Comment_(computer_programming)

Acknowledgments

None so far.

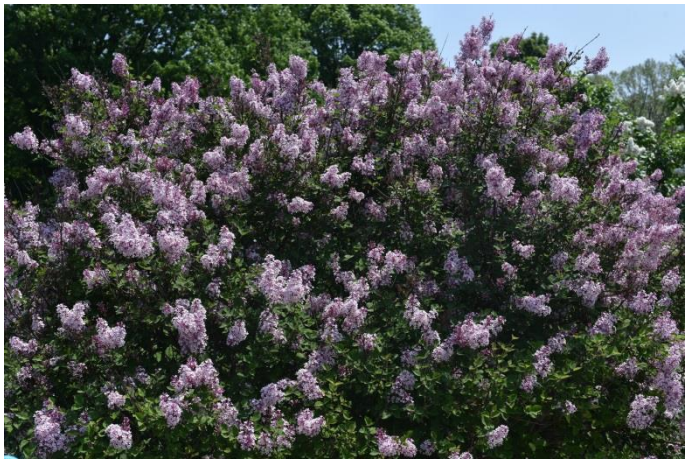
Colophon

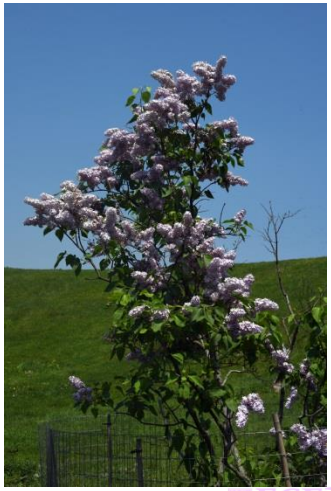
These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

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.