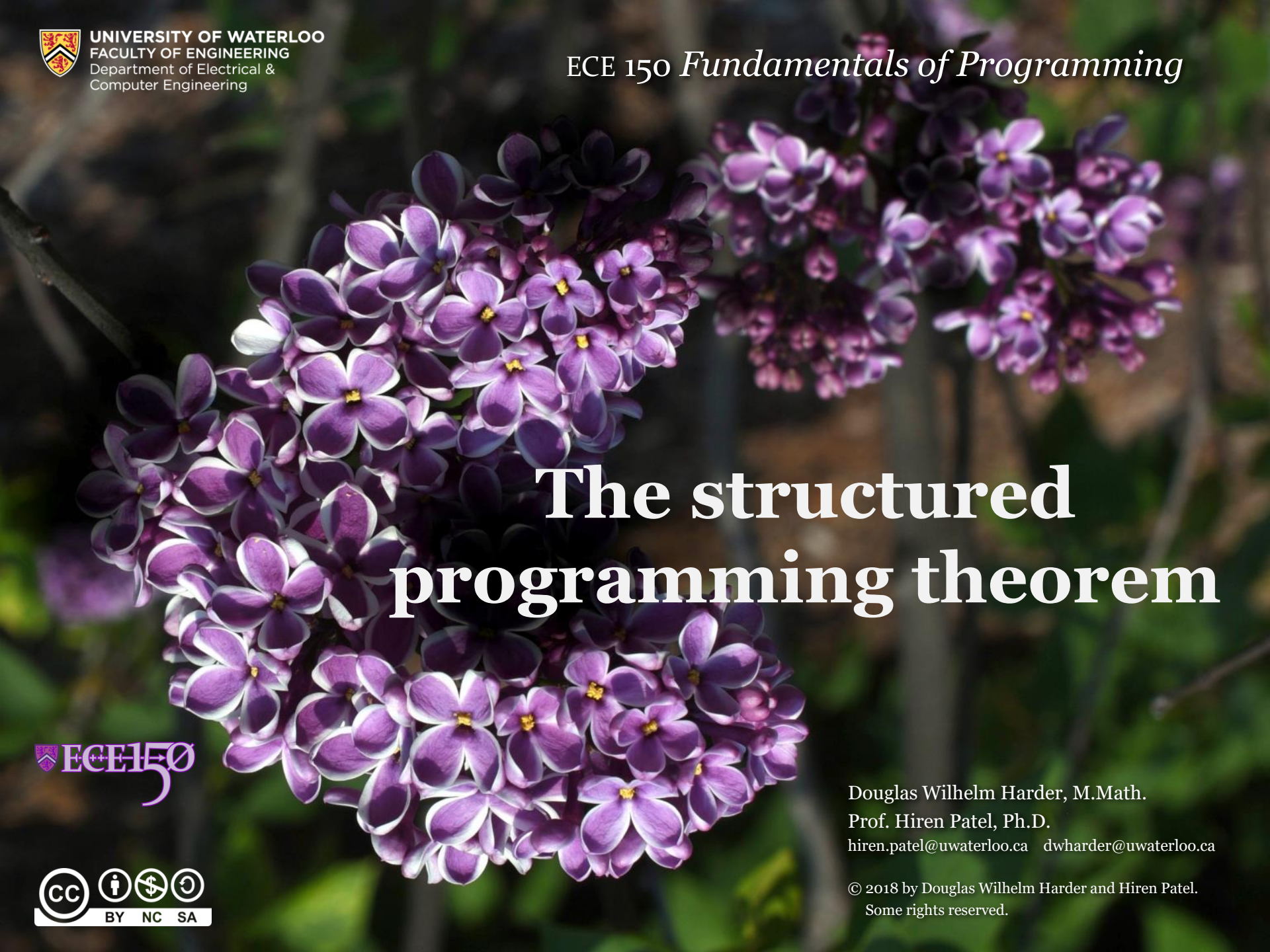
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

ECE 150 *Fundamentals of Programming*



The structured programming theorem

ECE150

Douglas Wilhelm Harder, M.Math.
Prof. Hiren Patel, Ph.D.
hiren.patel@uwaterloo.ca dwharder@uwaterloo.ca

Outline

- In this lesson, we will:
  - Review theorems from secondary school
  - Review the statements we have seen to this point
  - Look at programs using *goto* statement
  - Describe some solvable and unsolvable problems
  - Present the structured programming theorem
  - Discuss how to use the structured programming theorem when writing programs to solve problems

Background

- You've seen used and observed numerous applications, some of which may amaze you
  - You ask yourself, how can you do something like this?
  - The answer: Libraries and the *structured programming theorem*

Joust, Williams Electronics

Fortnite, Epic Games

Background

- The C++ language is very complex, and for many and good reasons
  - There are many features designed to support the development of software
  - The C++ standard costs 198 Swiss Francs (~400 CAD) and is over 1600 pages
  - Most of you will never master the entire language
    - That's okay, no instructor has, either…

Background

- In your mathematics courses, you have seen a few theorems:
  - A theorem is a statement must be true if the given premises are true

- Examples:
  - Euclid's theorem

    There are infinitely many prime numbers
  - The Pythagorean theorem

    If a, b, and c are sides of a right-angled triangle, and c is the side opposite the right angle, then $a^2 + b^2 = c^2$
  - The remainder theorem

    If a polynomial $p(x)$ is divided by $(x - r)$, the remainder is $p(r)$
  - The factor theorem

    A polynomial $p(x)$ has a factor $(x - r)$ if and only if $p(r) = 0$

The flow charts so far

- To this point, we have described:
  - Conditional statements
  - Repetition statements
    - For loops and while loops

- You may be aware of:
  - Functions
  - Arrays
  - Object-oriented programming

Equivalent implementations

- You may be aware of the goto statement
  - It exists in C++

```
while ( capacity > 0 ) {
    int n{0};
    int k{1};
    double max{array[0]};

    while ( k < capacity ) {
        if ( max < array[k] ) {
            array[k - 1] = max;
            max = array[k];
        } else {
            array[k - 1] = array[k];
            n = k;
        }

        ++k;
    }

    array[k - 1] = max;
    capacity = n;
}
```

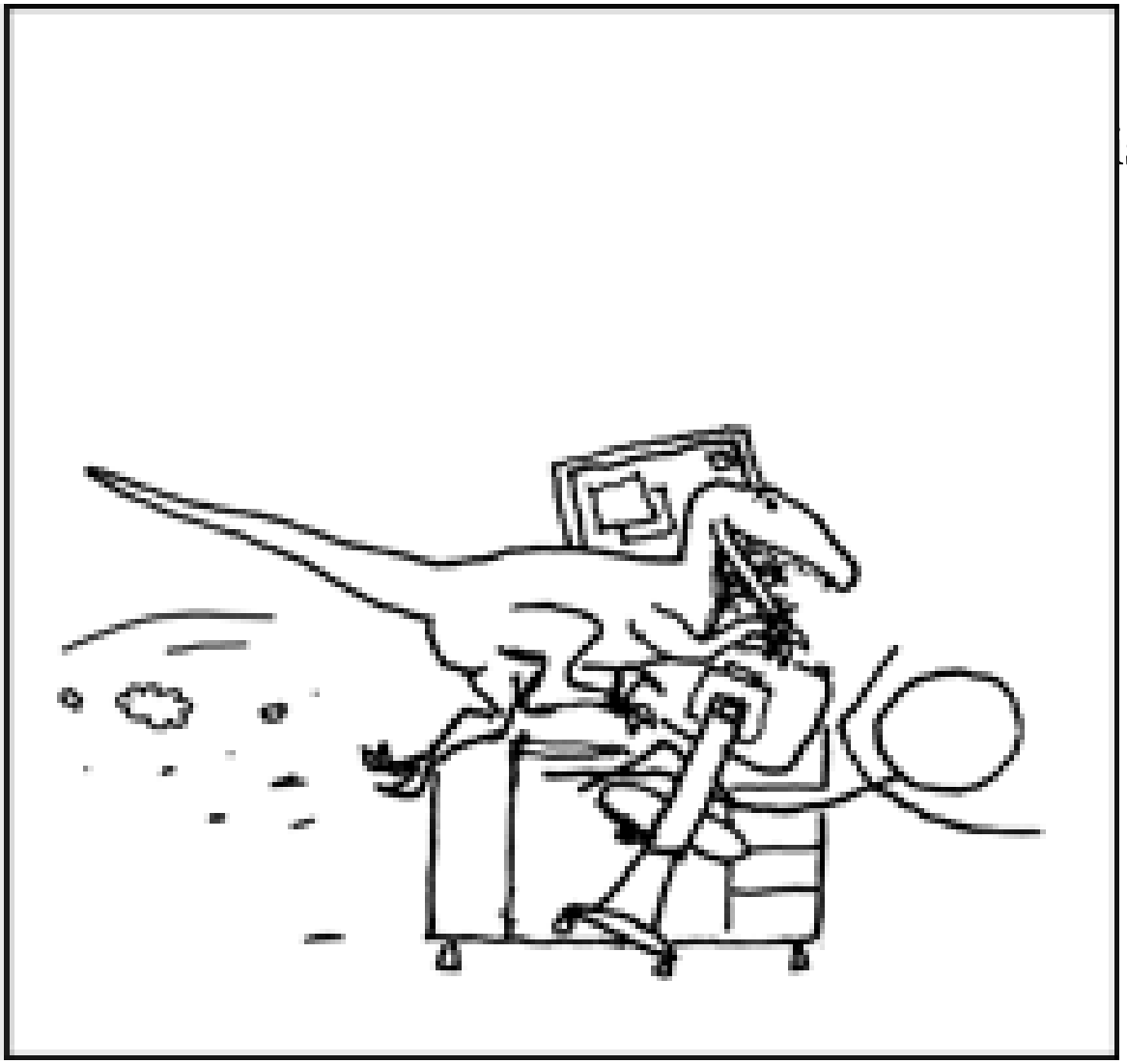
```
start: if ( capacity == 0 ) {
    return;
}

int n{0};
int k{1};
double max{array[0]};

next: if ( k == capacity ) {
    array[k - 1] = max;
    capacity = n;
    goto start;
}

if ( max < array[k] ) {
    array[k - 1] = max;
    max = array[k];
} else {
    array[k - 1] = array[k];
    n = k;
}

++k;
goto next;
```

It's rocket science…

- Even th s rule:

R

Th



Solvable problems

- What problems can be solved?
  - Finding the least expensive path between two locations
  - Given a set of packages with given volumes, masses, and profit, and given a truck that can hold a fixed mass and volume,

    what is the optimal selection of packages to carry?
  - In visiting *n* cities,

    what is the least expensive sequence in which to visit them?
  - Given *n* integers,

    is there a subset of these integers that sum to zero?
  - Given *n* people of which everyone is known to someone else, but no one is known to everyone,

    what is the smallest subset of these people so that everyone knows at least one person this subset?

Unsolvable problems

- What problems cannot be solved?
  - Given a program with a particular set of inputs, will the program go into an infinite loop?
  - Sometimes the answer is obvious:

```
int main() {
    int n{};
    std::cin >> n;
    int sum{0};

    for ( int k{0}; k < 10; k += n ) {
        sum += k;
    }

    std::cout << sum << std::endl;

    return 0;
}
```

The structured programming theorem

- Fortunately, the following theorem says we don't need any more:
  - Any program that can be written can be written using conditional and repetition statements

The structured programming theorem

- Consequences:
  - You understand most of the tools already
  - When you learn a new programming language,
    first understand how to write:
    - Conditional statements
    - While loops
- For example, consider MATLAB

```
if n > 0
    % do something
end
```

```
while n > 0
    % do something
end
```

```
for k = 1:10
    % do something
end
```

- When you tackle a problem that require the repetition of a given set of operations, think in terms of a while loop:

  "We must perform this set of instructions while this condition is true"

Moral of the story…

- You will not be required to investigate the structured programming theorem

- Use this as a guide to authoring programs:
  - Always think in terms of looping while some condition is true
  - Understand when you need to terminate the loop
  - Always ask Boolean-valued questions,
    either for while loops or conditional statements

Summary

- Following this lesson, you now:
  - Understand that you should never use a goto statement
  - Understand the consequences of the structured programming theorem
    - All programs that can be written can be written using only conditional and repetition statements
  - When trying to write a program to solve a problem:
    - Always think in terms of executing a body of statements if or while some condition is true

References

[1] Wikipedia
https://en.wikipedia.org/wiki/Structured_program_theorem

Acknowledgements

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.