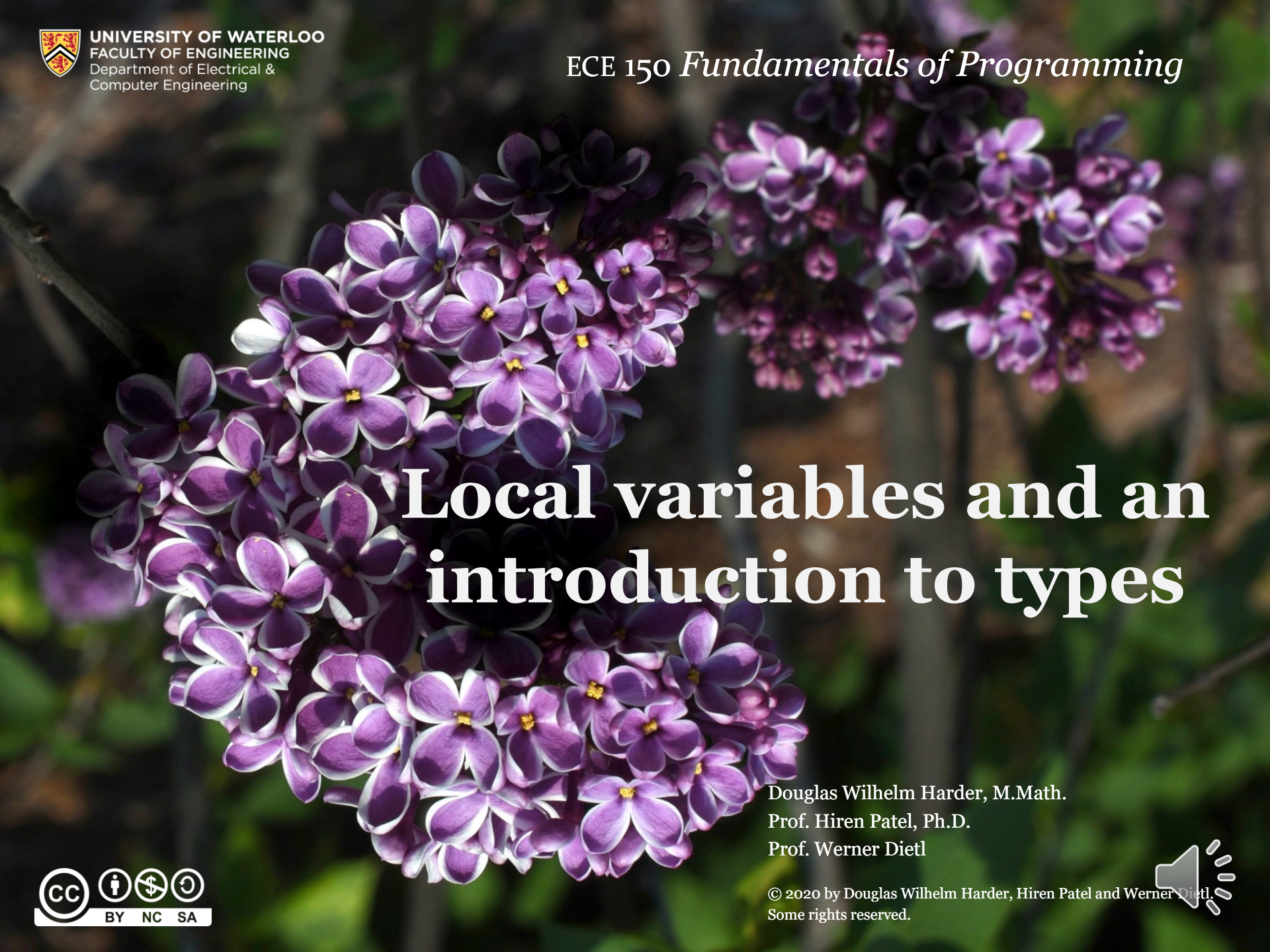
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

ECE 150 *Fundamentals of Programming*



Local variables and an introduction to types

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

Outline

- In this lesson, we will:
  - Introduce the need for temporary memory storage
  - Show how to get user input
  - Use the locally stored data in output
  - Introduce appropriate terminology

Background

- Literal data must be hard-coded into the program before it is compiled
  - This is useless if the program is to react to different input
  - For now, we will access data from the keyboard
    - Data can also be accessed from sensors and through other communication networks

- Data will be stored in *local variables*
  - We will start by seeing how we can temporarily store:
    - Integers
    - Characters
    - Strings
    - Floating-point numbers
    - Boolean

Integers

- The following program allows the user to enter an integer:

```
#include <iostream>

// Function declarations
int main();

// Function definitions
int main() {
    // Local variable declaration
    int n;

    std::cout << "Enter an integer: ";
    std::cin >> n;
    std::cout << "You entered " << n << std::endl;

    return 0;
}
```

Integers

- The line

```
// Local variable declaration
int n;
```

  says to the compiler:
  - The identifier `n` will be used to temporarily store an integer value
  - It is referred to as a *local variable*

- The identifier `int` is a *keyword*:
  - This is reserved by C++ to allow you to specify what an identifier can store
  - The keyword `int` describes the *type* of the local variable
    - In this case, an integer
  - We are *declaring* `n` to be of type `int`
    - A variable must be declared before it is used

Integers

- The line

```
std::cin >> n;
```

  says to the compiler:
  - Call an appropriate subroutine to wait for the user to enter an integer and assign that value to n

```
// Function definitions
int main() {
    // Local variable declaration
    int n;

    std::cout << "Enter an integer: ";
    std::cin >> n;
    std::cout << "You entered " << n << std::endl;

    return 0;
}
```

Integers

- The line

```
std::cout << "You entered " << n << std::endl;
```

  says to the compiler:
  - Print the string "`You entered `"
  - Print the value stored in the local variable `n`
  - Go the start of the next line

```
// Function definitions
int main() {
    // Local variable declaration
    int n;

    std::cout << "Enter an integer: ";
    std::cin >> n;
    std::cout << "You entered " << n << std::endl;

    return 0;
}
```

Integers

- When you execute this program, you see:

```
Enter an integer: 42
You entered 42
```

```
// Function definitions
int main() {
    // Local variable declaration
    int n;

    std::cout << "Enter an integer: ";
    std::cin >> n;
    std::cout << "You entered " << n << std::endl;

    return 0;
}
```

Characters

- The following program allows the user to enter a character:

```
#include <iostream>

// Function declarations
int main();

// Function definitions
int main() {
    // Local variable declaration
    char ch;

    std::cout << "Enter a character: ";
    std::cin >> ch;
    std::cout << "You entered '" << ch << << "'" << std::endl;

    return 0;
}
```

Characters

- The line

```
char ch;
```

  says to the compiler:
  - The identifier `ch` will be used to temporarily store a character

- The identifier `char` is a keyword:
  - The type of `ch` is `char`

Strings

- The following program allows the user to enter a string:

```
#include <iostream>
#include <string>

// Function delcarations
int main();

// Function definitions
int main() {
    // Local variable declaration
    std::string given_name;

    std::cout << "Enter your given name: ";
    std::cin >> given_name;
    std::cout << "Hi " << given_name << "!" << std::endl;

    return 0;
}
```

Strings

- The line

```
std::string given_name;
```

  says to the compiler:
  - The identifier `given_name` will be used to temporarily store an instance of the *string class*

- The symbol `std::string` consists of a the namespace `std` and the identifier `string`
  - The type of `given_name` is `std::string`

Floating-point numbers

- The following program allows the user to enter a float:

```
#include <iostream>

// Function declarations
int main();

// Function definitions
int main() {
    // Local variable declaration
    double expected_salary;

    std::cout << "Enter your expected salary: ";
    std::cin >> expected_salary;
    std::cout << "Your expected salary is " << expected_salary
              << std::endl;

    return 0;
}
```

Floating-point numbers

- The line

  ```
  double expected_salary;
  ```

  says to the compiler:
  - The identifier `expected_salary` will be used to temporarily store a floating-point number

- The identifier `double` is a keyword:
  - The type of `expected_salary` is `double`

- The name `double` is short for

  double-precision floating-point number

Boolean values

- The following program allows the user to enter a Boolean value:
  - The entered value must be `0` or `1`
  - The Boolean literals `true` and `false` are not allowed for input

```
#include <iostream>

// Function declarations
int main();

// Function definitions
int main() {
    // Local variable declaration
    bool does_enjoy_monty_python;

    std::cout << "Do you enjoy Monty Python? (0 or 1)? ";
    std::cin >> does_enjoy_monty_python;
    std::cout << "Are you sick? " << does_enjoy_monty_python
              << std::endl;

    return 0;
}
```

Boolean values

- The line

```
bool does_enjoy_monty_python;
```

  says to the compiler:
  - The identifier `does_enjoy_monty_python` is declared to be of type `bool`
    - A Boolean value

Types

- The type tells the compiler how the value is to be stored in memory
  - We will see later how these values are stored
  - For example, the type `int` allows you to store any integer from `-2147483648` up to `2147483647`

Additional comments

- These temporary local values are lost when the function `main()` exits
  - Later, we will see how we can modify and use local variables
  - We will also discuss when we can use local variables, or their *scope*

Summary

- Following this lesson, you now:
  - Understand the need for local variables
  - Know that each local variable
    - Has a type
    - Must be declared before it is used
  - Know how to get data from they keyboard
  - Know how to refer to the value stored in a local variable

References

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

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.