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

ECE 150 *Fundamentals of Programming*

Assignment operators

ECE150

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

Outline

- In this lesson, we will:
  - Look at swapping two local variables
  - Describe the automatic assignment operators
  - Describe the automatic increment and decrement operators

Background

- We have seen that assignment can be used for local variables:

```
double x{};
double y{};

std::cout << "Enter a value of x: ";
std::cin >> x;

y = x*x + 3.2*x - 1.5;

std::cout << y << std::endl;
```

Swapping values

- Suppose we have two local variables and we need to swap their values:

```
int main() {
    double x{};
    double y{};
    std::cin > x;
    std::cin > y;

    // Make sure x >= y
    if ( x < y ) {
        // Swap x and y
    }

    std::cout << "max( x, y ) = " << x << std::endl;

    return 0;
}
```

Swapping values

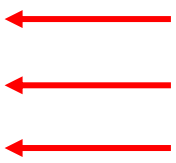
- Problem: Once a local variable is assigned, it original value is overwritten:

```
// Make sure x >= y
if ( x < y ) {
    x = y;
    // Both 'x' and 'y' have the same value
    //  - the original value of 'x' is lost
}
```

Swapping values

- Solution: Save the value of x in a temporary local variable before assigning it the value of y

```
// Make sure x >= y
if ( x < y ) {
    double tmp{x};
    x = y;
    y = tmp;
}

std::cout << "max(x, y) = " << x << std::endl;
```



tmp	5.4
y	18.3 → 5.4
x	18.3 → 18.3

Automatic assignment

- Another common collection of assignments is to update a variable:

```
x = x + 3;
x = x - 10;
x = x*2;
x = x/(r*r + 2*r + 1);
n = n%10;
```

- These can instead be written as:

```
x += 3;                 // automatic addition
x -= 10;                // automatic subtraction
x *= 2;                 // automatic multiplication
x /= r*r + 2*r + 1;     // automatic division
n %= 10;                // automatic modulus
                        // automatic remainder
```

Autoincrement and autodecrement

- Another very common operation is to change an integer local variable by plus or minus one:

```
n = n + 1;
n = n - 1;
```

- These can instead be written as:

```
n += 1;              // auto-addition of 1
n -= 1;              // auto-subtraction by 1
```

- These can instead be written as:

```
++n;
n++;          // auto-increment
--n;
n--;          // auto-decrement
```

What's the difference?

- What is the difference between ++n and n++ ?
  - Remember that `a + b` evaluates to whatever the sum is, so

```
x = a + b;   // Assign x the sum of a + b
```

  - Both ++n and n++ add one to n, but:

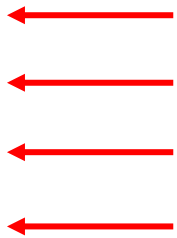
    What does ++n and n++ evaluate to?

What's the difference?

```
#include <iostream>

// Function declarations
int main();

// Function definitions
int main() {
    int n{752};

    std::cout << (++n  ) << std::endl;
    std::cout << (  n  ) << std::endl;
    std::cout << (  n++) << std::endl;
    std::cout << (  n  ) << std::endl;

    return 0;
}
```

Output:

753
753
753
754



What's the difference?

- If used by itself, as a single statement, both `++n;` and `n++;`
  do exactly the same thing with no difference what-so-ever
  - They both add one to `n`

- It is only if you do something with what these evaluate to that it makes a difference…
  - You will never use this in this course

- One small difference: `++n` is more efficient than `n++`, because the latter must temporarily store the original value of `n`
  - In this class, we will always use `++n;` and `--n;` as a single statement

Summary

- Following this lesson, you now:
  - Know how to swap local variables
  - Understand

    ```
    x = x + y*y + 1;
    ```

    and

    ```
    x += y*y + 1;
    ```

    are equivalent
  - Understand the automatic assignment operators

    ```
    -=   *=   /=   %=
    ```
  - Know the auto-increment and auto-decrement operators

    ```
    ++n     n++     --n     n--
    ```

References

[1] No references?

Acknowledgements

- Allen Du for noting the missing parentheses on Slide 10.

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.