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

ECE 150 *Fundamentals of Programming*

Break statements

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

© 2018-20 by Douglas Wilhelm Harder and Hiren Patel.
Some rights reserved.

BY NC SA

Outline

- In this lesson, we will:
  - Introduce the concept of breaking out of a loop
  - Modify a previous example to finish quicker
  - Look at how to break out of nested loops

BY NC SA

Determining if an integer is prime

- Consider this program:

```
int main() {
    int n{};
    std::cout << "Enter an integer: ";
    std::cin >> n;

    bool is_prime{true};

    if ( n%2 == 0 ) {
        is_prime = false;
    } else {
        for ( int k{3}; k < n; k += 2 ) {
            if ( n%k == 0 ) {
                is_prime = false;
            }
        }
    }
```

CC BY NC SA

Determining if an integer is prime

```
    if ( is_prime ) {
        std::cout << "The integer " << n << " is prime" << std::endl;
    } else {
        std::cout << "The integer " << n << " is not prime" << std::endl;
    }

    return 0;
}
```

BY NC SA

Ending a loop early

- Suppose you test if 303 is prime:

```
303%  2 == 1
303%  3 == 0
303%  5 == 3
303%  7 == 2
303%  9 == 6
303% 11 == 6
303% 13 == 4
          ⋮
303%301 == 2
```

  – After $k = 3$, we're done; we know that 303 is not prime…
    - So why test all other numbers?

Ending a loop early

- The `break` statement allows us to terminate a loop:
  - The loop immediate stops:
    - The condition is not tested
    - The update statement is not executed
  - Execution jumps to the end of the loop and continues from there

```
if ( n%2 == 0 ) {
    is_prime = false;
} else {
    for ( int k{3}; k < n; k += 2 ) {
        if ( n%k == 0 ) {
            is_prime = false;
            break;
        }
    }
    // Execution continues here...
}
```

  - This is useful, because once any number divides *n*,
    we no longer have to run any more tests

CC BY NC SA

Ending early if it is prime

- Now, given an integer n that is not prime, if $n = m_1 m_2$ then either:
  1. If n is a perfect square, it may be that $m_1 = m_2 = \sqrt{n}$,
  2. Otherwise, if $m_1 < \sqrt{n}$, then $m_2 > \sqrt{n}$

- For example, $\sqrt{420} \approx 20.4939$
- We see that,

$$
\begin{aligned}
420 &= 2 \times 210 &&= 3 \times 140 \\
&= 4 \times 105 &&= 5 \times 84 \\
&= 6 \times 70 &&= 7 \times 60 \\
&= 10 \times 42 &&= 12 \times 35 \\
&= 14 \times 30 &&= 15 \times 28 \\
&= 20 \times 21
\end{aligned}
$$

BY NC SA

Ending early if it is prime

- Thus, an even better program is:

```
if ( n%2 == 0 ) {
    is_prime = false;
} else {
    for ( int k{3}; k < n; k += 2 ) {
        if ( n%k == 0 ) {
            is_prime = false;
            break;
        }

        //           ___
        // If k > \/ n  and n%k == 0, then n/k < k, so we would
        // have already tested it. Thus, we only need to test
        // those k less than or equal to the square root of n.
        if ( k*k > n ) {
            break;
        }
    }
    // Execution continues here...
}
```

BY NC SA

Ending a loop early

- Consider the benefits:
  - To test if a number around 1 000 000 is prime:
    - Previously, we tested approximately 500 000 numbers
    - Now we test at approximately 500
  - To test if a number around 100 million is prime:
    - Previously, we would have tested approximately 50 million numbers
    - Now we test at approximately 5000

BY NC SA

Modifying the condition

- As a simple observation, we didn't have to use a break statement, as both these conditions could have be added to the condition

```
if ( n%2 == 0 ) {
    is_prime = false;
} else {
    //                                                    ___
    // Stop looping if we ever find is_prime == false or k > \/ n
    for ( int k{3}; is_prime && (k*k <= n); k += 2 ) {
        if ( n%k == 0 ) {
            is_prime = false;
        }
    }
}
```

  – Both work, both are acceptable approaches to solving this problem

CC BY NC SA

Finding a sum of squares

- Suppose we want to find if n is the sum of two non-zero squares:
  - Is $n = m_1^2 + m_2^2$ for two non-zero integers m_1 and m_2?
  - For many engineering problems, we only need to have one example
    - Thus, once we find one pair of integers, we're finished…

BY NC SA

Finding a sum of squares

- Consider this program:

```
int main() {
    int n{};
    std::cout << "Enter an integer: ";
    std::cin >> n;

    bool is_found{false};

    for ( int m1{1}; m1*m1 < n; ++m1 ) {
        for ( int m2{1}; m1*m1 + m2*m2 <= n; ++m2 ) {
            if ( (m1*m1 + m2*m2) == n ) {
                is_found = true;
                break;
            }
        }
    }

    // What are m1 and m2?
    return 0;
}
```

BY NC SA

Finding a sum of squares

- Consider this program:

```
int main() {
    int n{};
    std::cout << "Enter an integer: ";
    std::cin >> n;

    bool is_found{false};

    int m1{};
    int m2{};

    for ( m1 = 1; m1*m1 < n; ++m1 ) {
        for ( m2 = 1; m1*m1 + m2*m2 <= n; ++m2 ) {
            if ( (m1*m1 + m2*m2) == n ) {
                is_found = true;
                break;
            }
        }
    }
```

BY NC SA

Finding a sum of squares

```
    if ( is_found ) {
        std::cout << n << " = " << m1 << "^2 + "
                                << m2 << "^2" << std::endl;
    } else {
        std::cout << n << " is not the sum of two non-zero squares"
                  << std::endl;
    }

    return 0;
}
```

Finding a sum of squares

- We now try running our program:

```
Enter an integer: 121
121 is not the sum of two non-zero squares

Enter an integer: 122
122 = 12^2 + 1^2
```

BY NC SA

Finding a sum of squares

- The break only exits the inner loop

```
for ( m1 = 1; m1*m1 < n; ++m1 ) {
    for ( m2 = 1; m1*m1 + m2*m2 <= n; ++m2 ) {
        if ( (m1*m1 + m2*m2) == n ) {
            is_found = true;
            break;
        }
    }
    // The break jumps to this point...
}
```

BY NC SA

Finding a sum of squares

- Solution
  - If we find a pair of integers that satisfies our condition, break out of the outer loop, as well

```
for ( m1 = 1; m1 < n; ++m1 ) {
     for ( m2 = m1; m2 < n; ++m2 ) {
         if ( (m1*m1 + m2*m2) == n ) {
             is_found = true;
             break;   // exits the inner for loop
         }
     }

     if ( is_found ) {
         break;   // exits the outer for loop
     }
 }
```

BY NC SA

Finding a sum of squares

- Does this work?

```
for ( m1 = 1; !is_found && (m1*m1 < n); ++m1 ) {
    for ( m2 = m1; !is_found && (m1*m1 + m2*m2 <= n); ++m2 ) {
        if ( (m1*m1 + m2*m2) == n ) {
            is_found = true;
        }
    }
}
```

BY NC SA

Summary

- Following this lesson, you now
  - Understand the purpose of a break statement
    - It ends the execution of a for loop
  - Know that the break statement is just `break;`
  - Understand how to finish a loop early if there is no need to continue executing the loop
    - It only jumps out of the loop in which it is found

BY NC SA

References

[1] Wikipedia
https://en.wikipedia.org/wiki/Control_flow#Early_exit_from_loops

[2] cplusplus.com
http://www.cplusplus.com/doc/tutorial/control/

Acknowledgments

Proof read by Dr. Thomas McConkey and Charlie Liu.

BY NC SA

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.

BY NC SA

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.

CC BY NC SA