

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


ECE 150 *Fundamentals of Programming*

Debugging

ECE150

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

© 2020 by the above. Some rights reserved.






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

Logging execution 2

Outline

- This is the fifth in a sequence of six topics on
 - C assertions
 - Code development strategies
 - Testing
 - Commenting your code
 - Using print statements for debugging
 - Using the debugger

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

Logging execution 3

Outline

- In this topic, we will:
 - Describe the purpose of the debugger
 - Look at how it is used
 - Explain some of the benefits

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

Logging execution 4

Purpose of debugging

- Logging can be frustrating,
 - as you must insert statements throughout your code
- The debugger does all this work for you:
 - It prepares a different executable: one that tracks all variables
 - We'll try this on a program that works
- The term *debugger* tends to scare students
 - We will start by showing how it works with correct code



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

Logging execution 5

Logging with high-low

- Here is a program that calculates the gcd of two integers:

```
#include <iostream>

// Function declarations
int main();
int gcd( int m, int n );

// Function definitions
int main() {
    int m{8*3 *7*11*13 *23};
    int n{4*9*5 *11 *17*23};
    // The gcd should be 4x3x11x23 = 3036

    std::cout << gcd( m, n ) << std::endl;

    return 0;
}

int gcd( int m, int n ) {
    if ( m == 0 ) {
        return n;
    } else if ( n == 0 ) {
        return m;
    }

    while ( n != 0 ) {
        int r{ m%n };
        m = n;
        n = r;
    }

    return m;
}
```

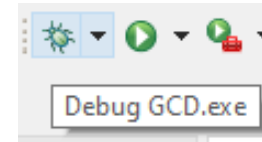


UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

Logging execution 6

Logging with high-low

- In Eclipse (or whichever IDE you are using), you can execute this code in *debug mode*



- Why “bugs”?
 - The term was used to describe technical issues since the mid 1800s
 - On September 9th, 1947, operators found an actual bug (a moth) in a computer, and labeled it as “first actual case of bug being found”
 - RAdm. Grace Hopper popularized the use of “bugs” and “debugging” in computer hardware and software development

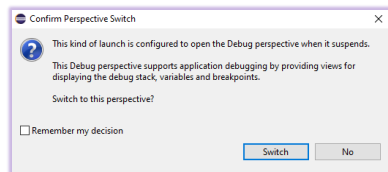


UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

Logging execution 7

Logging with high-low

- When you *debug* your program, you are asked:



- Please do switch

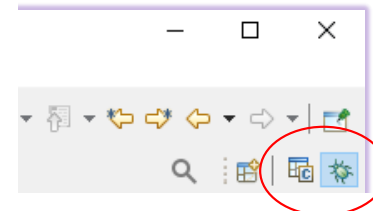


UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

Logging execution 8

Logging with high-low

- For C/C++ development, Eclipse offers two primary perspectives:
 - C/C++
 - Debug



Logging execution 9

Starting the debugger

The statement to be executed next.

All local variables and parameters, their types, and their values of the currently executing function

Name	Type	Value
op-m	int	0
op-n	int	0

Logging execution 10

Relevant controls

Stop debugging

Stepping controls

Logging execution 11

Logging with high-low

Step Into

Step Return

Step Over

Logging execution 12

Step Into

- The Step Into (F5)
 - Executes the next statement, unless there is one or more function calls
 - If there is one or more function calls in the next statement, step into executing the next function call
 - Note, it will not step into any function call in the standard library, only your functions you authored in Eclipse





Step Over

- The Step Over (F6)
 - Executes the next statement
 - All function calls are made, but you simply see the result
 - If there are no function calls, Step Over is equivalent to Step Into



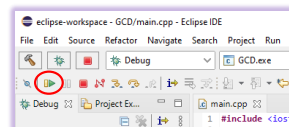
Step Return

- The Step Return (F6)
 - Finishes executing the current function and returns to the statement that contains the call to this function
 - Thus button is inactive when in main()



Stepping through code

- The Resume button continues execution until either:
 - The program finishes
 - A breakpoint is reached
 - ...and a few other features



Stepping through code

```

1  main.cpp
2
3  int main();
4  int gcd( int m, int n );
5
6  int main() {
7      int m[8*3 *7*11*13 *23];
8      int n[4*5*7 *11 *17*23];
9      // The gcd should be 493*13*23 = 3036
10
11     std::cout << gcd( m, n ) << std::endl;
12
13 }
14
15
16 int gcd( int m, int n ) {
17     if ( m == 0 ) {
18         return n;
19     } else if ( n == 0 ) {
20         return m;
21     }
22
23     while ( n != 0 ) {
24         int r[ m% n ];
25         m = n;
26         n = r;
27     }
28
29     return m;
30 }
31

```

Name	Type	Value
m	int	0
n	int	0



UNIVERSITY OF WATERLOO
 Faculty of Engineering
 Department of Electrical and Computer Engineering
Logging execution 17
Stepping through code

```

main.cpp
2
3 int main();
4 int gcd(int m, int n);
5
6 int main() {
7     int m[8*3  *7*11*13  *23];
8     int n[4*9*5  *11  *17*23];
9     // The gcd should be 4x3x11x23 = 3036
10
11     std::cout << gcd(m, n) << std::endl;
12     return 0;
13 }
14
15
16 int gcd(int m, int n) {
17     if ( m == 0 ) {
18         return n;
19     } else if ( n == 0 ) {
20         return m;
21     }
22
23     while ( n != 0 ) {
24         int r( m%n );
25         m = n;
26         n = r;
27     }
28     return m;
29 }
30
31
    
```

Name	Type	Value
00-m	int	52552
00-n	int	0



UNIVERSITY OF WATERLOO
 Faculty of Engineering
 Department of Electrical and Computer Engineering
Logging execution 18
Stepping through code

```

main.cpp
2
3 int main();
4 int gcd(int m, int n);
5
6 int main() {
7     int m[8*3  *7*11*13  *23];
8     int n[4*9*5  *11  *17*23];
9     // The gcd should be 4x3x11x23 = 3036
10
11     std::cout << gcd(m, n) << std::endl;
12     return 0;
13 }
14
15
16 int gcd(int m, int n) {
17     if ( m == 0 ) {
18         return n;
19     } else if ( n == 0 ) {
20         return m;
21     }
22
23     while ( n != 0 ) {
24         int r( m%n );
25         m = n;
26         n = r;
27     }
28     return m;
29 }
30
31
    
```

Name	Type	Value
00-m	int	52552
00-n	int	774180



UNIVERSITY OF WATERLOO
 Faculty of Engineering
 Department of Electrical and Computer Engineering
Logging execution 19
Stepping through code

```

main.cpp
2
3 int main();
4 int gcd(int m, int n);
5
6 int main() {
7     int m[8*3  *7*11*13  *23];
8     int n[4*9*5  *11  *17*23];
9     // The gcd should be 4x3x11x23 = 3036
10
11     std::cout << gcd(m, n) << std::endl;
12     return 0;
13 }
14
15
16 int gcd(int m, int n) {
17     if ( m == 0 ) {
18         return n;
19     } else if ( n == 0 ) {
20         return m;
21     }
22
23     while ( n != 0 ) {
24         int r( m%n );
25         m = n;
26         n = r;
27     }
28     return m;
29 }
30
31
    
```

Name	Type	Value
00-m	int	52552
00-n	int	774180



UNIVERSITY OF WATERLOO
 Faculty of Engineering
 Department of Electrical and Computer Engineering
Logging execution 20
Stepping through code

```

main.cpp
2
3 int main();
4 int gcd(int m, int n);
5
6 int main() {
7     int m[8*3  *7*11*13  *23];
8     int n[4*9*5  *11  *17*23];
9     // The gcd should be 4x3x11x23 = 3036
10
11     std::cout << gcd(m, n) << std::endl;
12     return 0;
13 }
14
15
16 int gcd(int m, int n) {
17     if ( m == 0 ) {
18         return n;
19     } else if ( n == 0 ) {
20         return m;
21     }
22
23     while ( n != 0 ) {
24         int r( m%n );
25         m = n;
26         n = r;
27     }
28     return m;
29 }
30
31
    
```

Name	Type	Value
00-m	int	52552
00-n	int	774180

```

Name : m
Details:52552
Default:52552
Decimal:52552
Hex:006668
Binary:1000011011001101000
Octal:02067150
    
```



UNIVERSITY OF WATERLOO
 Faculty of Engineering
 Department of Electrical and Computer Engineering
Logging execution 21
Stepping through code

```

main.cpp
2
3 int main();
4 int gcd(int m, int n);
5
6@ int main() {
7     int m{8*3} *7*11*13 *23;
8     int n{4*9*5} *11 *17*23;
9     // The gcd should be 4x3x11x23 = 3036
10
11     std::cout << gcd(m, n) << std::endl;
12
13     return 0;
14 }
15
16@ int gcd(int m, int n) {
17     if (m == 0) {
18         return n;
19     } else if (n == 0) {
20         return m;
21     }
22
23     while (n != 0) {
24         int r{m%n};
25         m = n;
26         n = r;
27     }
28
29     return m;
30 }
31
    
```

Name	Type	Value
@m	int	0
@n	int	774180

```

Name : m
Details:552552
Default:552552
Decimal:552552
Hex:0x86668
Binary:10000110111001101000
Octal:02067150
    
```



UNIVERSITY OF WATERLOO
 Faculty of Engineering
 Department of Electrical and Computer Engineering
Logging execution 22
Stepping through code

```

main.cpp
2
3 int main();
4 int gcd(int m, int n);
5
6@ int main() {
7     int m{8*3} *7*11*13 *23;
8     int n{4*9*5} *11 *17*23;
9     // The gcd should be 4x3x11x23 = 3036
10
11     std::cout << gcd(m, n) << std::endl;
12
13     return 0;
14 }
15
16@ int gcd(int m, int n) {
17     if (m == 0) {
18         return n;
19     } else if (n == 0) {
20         return m;
21     }
22
23     while (n != 0) {
24         int r{m%n};
25         m = n;
26         n = r;
27     }
28
29     return m;
30 }
31
    
```

Name	Type	Value
@m	int	0
@n	int	774180

```

Name : m
Details:0
Default:0
Decimal:0
Hex:0x0
Binary:0
Octal:0
    
```



UNIVERSITY OF WATERLOO
 Faculty of Engineering
 Department of Electrical and Computer Engineering
Logging execution 23
Stepping through code

```

main.cpp
2
3 int main();
4 int gcd(int m, int n);
5
6@ int main() {
7     int m{8*3} *7*11*13 *23;
8     int n{4*9*5} *11 *17*23;
9     // The gcd should be 4x3x11x23 = 3036
10
11     std::cout << gcd(m, n) << std::endl;
12
13     return 0;
14 }
15
16@ int gcd(int m, int n) {
17     if (m == 0) {
18         return n;
19     } else if (n == 0) {
20         return m;
21     }
22
23     while (n != 0) {
24         int r{m%n};
25         m = n;
26         n = r;
27     }
28
29     return m;
30 }
31
    
```

Name	Type	Value
@m	int	0
@n	int	774180

```

Name : m
Details:0
Default:0
Decimal:0
Hex:0x0
Binary:0
Octal:0
    
```



UNIVERSITY OF WATERLOO
 Faculty of Engineering
 Department of Electrical and Computer Engineering
Logging execution 24
Stepping through code

```

main.cpp
2
3 int main();
4 int gcd(int m, int n);
5
6@ int main() {
7     int m{8*3} *7*11*13 *23;
8     int n{4*9*5} *11 *17*23;
9     // The gcd should be 4x3x11x23 = 3036
10
11     std::cout << gcd(m, n) << std::endl;
12
13     return 0;
14 }
15
16@ int gcd(int m, int n) {
17     if (m == 0) {
18         return n;
19     } else if (n == 0) {
20         return m;
21     }
22
23     while (n != 0) {
24         int r{m%n};
25         m = n;
26         n = r;
27     }
28
29     return m;
30 }
31
    
```

Name	Type	Value
@m	int	552552
@n	int	774180

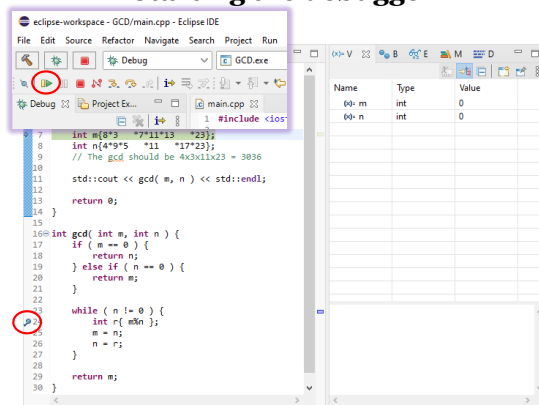
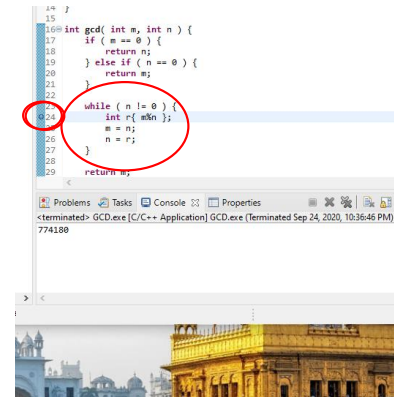
```

Name : m
Details:552552
Default:552552
Decimal:552552
Hex:0x86668
Binary:10000110111001101000
Octal:02067150
    
```





- At this point, you may have noticed a small weakness
 - Suppose you have a very large program
 - Do you really want to step through every statement until you get to the statements where you believe the code may be buggy?
 - The solution to this are breakpoints



- Following this lesson, you now:
 - Have been introduced to the debugger
 - Understand all it does is display the values of parameters and local variables
 - It allows you to execute one statement at a time
 - Know you can Step Into the execution of functions, or Step Over the execution of functions and just see the results
 - Are aware that you can even change the value of parameters and local variables on-the-fly while the debugger is executing
 - Know you can set break points that allow you to execute the program until the line in question is reached



References

- [1] Wikipedia:
<https://en.wikipedia.org/wiki/Debugger>



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.

