

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

ECE 150 *Fundamentals of Programming*

# Scope of local variables

CC BY NC SA

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.

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

Scope of local variables 2

## Outline

- In this lesson, we will:
  - Introduce the concept of scope for local variables
  - See the effect of using local variables outside of their scope
  - Explain why the scope of local variables is restricted
  - Look at a common mistake from first-year students

CC BY NC SA

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

Scope of local variables 3

## Accessing variables

- Consider this program:
 

```
#include <iostream>

// Function declarations
int main();

// Function definitions
int main() {
    for ( int k{4}; k < 7; ++k ) {
        std::cout << k << ", ";
    }

    // At this point, 'k' should equal 7
    std::cout << k << std::endl;

    return 0;
}
```

Desired output:  
4, 5, 6, 7

CC BY NC SA

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

Scope of local variables 4

## Accessing variables

- Problem:
 

```
example.cpp: In function 'int main()':
example.cpp:13:18: error: 'k' was not declared in this scope
std::cout << k << std::endl;
                  ^
```

  - The error message says that the identifier 'k' was not declared
  - However, it was declared in the initialization statement of the for loop, where we used it:
 

```
for ( int k{4}; k < 7; ++k ) {
    std::cout << k << ", ";
}
```

CC BY NC SA

UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF INFORMATION AND COMMUNICATIONS

Scope of local variables 5

## Accessing variables

- Consider this program:

```
#include <iostream>

// Function declarations
int main();

// Function definitions
int main() {
    int def_val(0);
    bool use_new();
    std::cout << "Do you want a value different from the default " << def_val << std::endl;
    std::cout << " (1 for yes or 0 for no)? ";
    std::cin >> use_new;

    if ( use_new ) {
        int new_val();
        std::cout << "Enter that value: ";
        std::cin >> new_val;
    }

    std::cout << "Your value is " << new_val << std::endl;

    return 0;
}
```

Desired output:  
Do you want a value different from the default 0  
(1 for yes or 0 for no)? 1  
Enter that value: 17  
Your value is 17



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF INFORMATION AND COMMUNICATIONS

Scope of local variables 6

## Repetition statements

- Problem:

example.cpp: In function 'int main()':  
example.cpp:20:38: error: 'new\_val' was not declared in this scope  
std::cout << "Your value is " << new\_val << std::endl;

- The error message says the identifier 'new\_val' was not declared

- However, it was declared and used in the conditional statement:

```
if ( use_new ) {
    int new_val();
    std::cout << "Enter that value: ";
    std::cin >> new_val;
}
```



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF INFORMATION AND COMMUNICATIONS

Scope of local variables 7

## The scope of a local variable

- To avoid confusion, a local variable can only be accessed from where it is defined to the end of the block in which it is defined

```
if ( some-condition ) {
    // Some code...

    bool is_found(false);

    // The local variable 'is_found' can only be accessed or
    // or assigned to up the end of this block
    // - Anywhere up to the matching closing brace of corresponding
    // opening brace of this conditional statement
}

// It cannot be accessed or assigned to here...
```



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF INFORMATION AND COMMUNICATIONS

Scope of local variables 8

## The scope of a local variable

- If this local variable is needed outside this conditional statement, it must be defined before the conditional statement

```
bool is_found();

if ( some-condition ) {
    // Some code...

    is_found = false;
    // The local variable 'is_found' can be accessed or
    // or assigned to up the end of this block
}

// It can be accessed or assigned to here, too
```





## The scope of a local variable

- That part of the program where a local variable can be accessed or assigned to is described as the scope of that local variable
  - If you try to access that local variable anywhere else, the compiler will indicate that that identifier is not in scope
- One common source of such errors are typos in the identifier names
  - If you declare a local variable `color_intensity` but then use the variable name `colour_intensity`, the second is an undeclared identifier, it is not in scope



## Scope of loop variables

- The scope of the loop variable is restricted to the for loop:

```
int main() {
```

```
    for ( int k{4}; k < 7; ++k ) {
        std::cout << k << ", ";
    }
```

Output:  
4, 5, 6,

```
    return 0;
}
```



## Scope of loop variables

- To access the loop variable, we must declare that variable outside the for loop:

```
int main() {
    int k{};
```

```
    for ( k = 4; k < 7; ++k ) {
        std::cout << k << ", ";
    }
```

Output:  
4, 5, 6, 7

```
    // At this point, 'k' is equal to 7
    std::cout << k << std::endl;
```

```
    return 0;
}
```



## Scope of loop variables

- This, while appearing awkward, is also valid:

```
int main() {
    int k{4}; // Initialize it here
```

```
    for ( k; k < 7; ++k ) {
        std::cout << k << ", ";
    }
```

Output:  
4, 5, 6, 7

```
    // At this point, 'k' is equal to 7
    std::cout << k << std::endl;
```

```
    return 0;
}
```



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF EDUCATION AS  
UNIVERSITY OF WATERLOO

Scope of local variables 13

## Example

- Consider this example

```
int main() {
    int count();
    std::cout << "How many values do you want to enter? ";
    std::cin >> count;
```

```
// The user must enter at least one value
if ( count < 1 ) {
    count = 1;
}
```

```
int maximum();
std::cin >> maximum;
```



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF EDUCATION AS  
UNIVERSITY OF WATERLOO

Scope of local variables 14

## Example

```
for ( int k{2}; k <= count; ++k ) {
    int tmp();
    std::cin >> tmp;

    if ( tmp > maximum ) {
        maximum = tmp;
    }
}
```

```
// The local variable 'maximum' is still in scope
std::cout << "The maximum number was " << maximum << std::endl;
```

```
return 0;
}
```

- We only require k and tmp in the loop
  - The reader knows that these variables will only be used here



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF EDUCATION AS  
UNIVERSITY OF WATERLOO

Scope of local variables 15

## Purpose of scope

- If you wanted, you could simply declare all local variables at the start of the function `int main()`

```
int main() {
    int count();
    int maximum();
    int k();
    int tmp();

    std::cout << "How many values do you want to enter? ";
    std::cin >> count;

    // ...

    return 0;
}
```



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF EDUCATION AS  
UNIVERSITY OF WATERLOO

Scope of local variables 16

## Purpose of scope

- Problems with this approach:
  - The reader of your code does not know which local variables are important and which are temporary
  - This makes it much more difficult to read through your code
    - With the local variables declared near the top, the reader can say, "These seem to be important..."
    - With the local variables declared in the for loop, the reader can say, "Ah, these are used here, and nowhere else..."



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF ENGINEERING AS  
UNIVERSITY OF WATERLOO

Scope of local variables 17

## Purpose of scope

- Also, if you declare all local variables at the start, you may
  - Accidentally use a local variable after it is declared but before it has any useful meaning
  - In a larger program, accidentally reuse a local variable when you should not



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF ENGINEERING AS  
UNIVERSITY OF WATERLOO

Scope of local variables 18

## Declaring local variables

- You can never declare a local variable twice in the same block
- In general, the same identifier should not be used for two different values, ever
  - If you were to declare an identifier to be a local variable twice, the *closest* is the one used:



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF ENGINEERING AS  
UNIVERSITY OF WATERLOO

Scope of local variables 19

## Declaring local variables

- For example:

```
int main() {
    int value{};
    std::cout << "Enter a positive value: ";
    std::cin >> value;

    if ( value <= 0 ) {
        std::cout << "The number " << value
            << " must be positive: ";
    }

    int value{};
    std::cin >> value;
    std::cout << "You just entered " << value << std::endl;
}

std::cout << "You entered " << value << std::endl;

return 0;
}
```

Output:

```
Enter a positive value: -5
The number -5 must be positive: 17
You just entered 17
You entered -5
```



UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
FACULTY OF ENGINEERING AS  
UNIVERSITY OF WATERLOO

Scope of local variables 20

## Summary

- Following this lesson, you now
  - Understand the scope of a local variable
  - Know the scope ends when the block in which the local variable is declared ends
  - Are aware that in general, local variables should be defined only when and where they are likely to be used
  - Understand that while a local variable cannot be declared twice in the same block of statements, the compiler allows the same identifier to be declared a second time in a block within the block
  - Realize that it is likely a bad idea to use the same identifier for two different local variables





## References

- [1] Wikipedia  
[https://en.wikipedia.org/wiki/For\\_loop](https://en.wikipedia.org/wiki/For_loop)
- [2] cplusplus.com  
<http://www.cplusplus.com/doc/tutorial/control/>



## 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.



## Acknowledgments

Proof read by Dr. Thomas McConkey and Charlie Liu.



## 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.

