

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

ECE 150 Fundamentals of Programming

Protecting pointers

BCEE150

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

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

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

Protecting pointers 2

Outline

- In this lesson, we will:
 - Understand how to protect pointers using const

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

Protecting pointers 3

Revisiting pointers

- A variable may be changed either directly or by changing what is at the corresponding address:

```
int main() {
    int x{42};
    int *p_x{&x};

    std::cout << *p_x << " == " << x << std::endl;

    // We can change the value of 'x' via 'p_x'
    *p_x = 91;
    std::cout << *p_x << " == " << x << std::endl;

    return 0;
}
```

Output:
42 == 42
91 == 91

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

Protecting pointers 4

Revisiting const

- A variable declared const cannot be reassigned:

```
int main() {
    int m{42};
    int const n{91};

    ++m;
    std::cout << "m == " << m << std::endl;
    ++n;
    std::cout << "n == " << n << std::endl;

    return 0;
}
```

Output:
example.cpp: In function 'int main()':
example.cpp:11:7: error: increment of read-only variable 'n'
++n;
^

Protecting pointers 5

Disallow changes to value of variable

```
int main() {
    int x{42};
    int y{33};
    // READ: Pointer to constant integer.
    int const *p_x{&x};

    // DISALLOWED: Change value of variable x via p_x.
    std::cout << *p_x << " : " << x << std::endl;
    *p_x = 22;
    std::cout << *p_x << " : " << x << std::endl;

    return 0;
}
```

Output: }
 pointer-const.cpp: In function 'int main()':
 pointer-const.cpp:14:10: error: assignment of read-only location '*p_x'
 *p_x = 22;

Protecting pointers 6

Disallow changes to address held in pointer

```
int main() {
    int x{42};
    int y{33};
    // READ: Constant pointer to integer.
    int *const p_x{&x};

    // DISALLOWED: Change the address held in p_x.
    std::cout << p_x << " : " << &x << std::endl;
    p_x = &y;
    std::cout << p_x << " : " << &y << std::endl;

    return 0;
}
```

Output: }
 pointer-const.cpp: In function 'int main()':
 pointer-const.cpp:13:9: error: assignment of read-only variable 'p_x'
 p_x = &y;

Protecting pointers 7

Disallow changes to address held in pointer and value of variable

```
Output:
pointer-const.cpp:13:9: error: assignment of read-only variable 'p_x'
  p_x = &y;
  ^
pointer-const.cpp:18:9: error: assignment of read-only location '*p_x'
  *p_x = 22;
  ^
// READ: Constant pointer to constant integer.
int const *const p_x{&x};

// DISALLOWED: Change the address held in p_x.
std::cout << p_x << " : " << &x << std::endl;
p_x = &y;
std::cout << p_x << " : " << &y << std::endl;

// DISALLOWED: Change the value of variable pointed to by p_x.
std::cout << p_x << " : " << &x << std::endl;
*p_x = 22;
std::cout << p_x << " : " << &x << std::endl;

return 0;
}
```

Protecting pointers 8

Reading these declarations

- Read the declaration
 int const *p_x{ ... };
 as saying "what is at the address p_x is constant"
- Read the declaration
 int *const p_x{ ... };
 as saying "the pointer p_x is constant"
- Finally, read the declaration
 int const *const p_x{ ... };
 as saying "the pointer p_x and what is at that address are both constant"

Protecting pointers 9

Addresses of constants

- This is necessary if you assign a pointer the address of a constant:

```
int main() {
    int const N{42};
    int *p_N{ &N };

    // This is not allowed: you cannot assign to 'N'
    //     N = 91;
    // Try assigning to 'N' indirectly
    *p_N = 91;

    return 0;
}
```

```
example.cpp: In function 'int main()':
example.cpp:3:15: error: invalid conversion from 'const int*' to 'int*'
    int *p_N{ &n };
                ^~
```



Protecting pointers 10

Addresses of constants

- You must declare the value of the point to be a constant

```
int main() {
    int const N{42};
    int const *p_N{ &N };

    // Neither of these are allowed
    //     N = 91;
    //     *p_N = 91;
    std::cout << *p_N << std::endl;

    return 0;
}
```

Output:
42



Protecting pointers 11

Summary

- Following this lesson, you now
 - Understand how to protect pointers



Protecting pointers 12

References

- [1] No references?





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.



CC BY-NC-SA

BY



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

ECE150