

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

ECE 150 *Fundamentals of Programming*

Lifetime of dynamically allocated memory

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

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

CC BY NC SA

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

Lifetime of dynamically allocated memory 2

Outline

- In this lesson, we will:
 - Discuss the lifetime of dynamically allocated memory
 - Author and discuss some functions that allocate, access and manipulate such memory
 - We will make changes that cannot be done with statically allocated memory

CC BY NC SA

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

Lifetime of dynamically allocated memory 3

Lifetime

- Note that dynamically allocated memory remains allocated until:
 - The program terminates
 - The memory is deallocated with `delete` or `delete[]`
- This means that such memory is still accessible even after a function returns
 - The problem is that we must keep our hands on the address
 - If we lose the address, we lose the memory

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

Lifetime of dynamically allocated memory 4

Allocating instances of a type

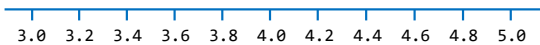
- We will author three functions:
 - The first allocates an array of the required size
 - The second allows the user to make a change to the array
 - The third cleans up the array and deallocates it

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

Lifetime of dynamically allocated memory 5

Allocate and initialize

- The first function will:
 - Allocate an array of the required capacity
 - It will take two additional arguments a and b
 - It will initialize the array so that $\text{array}[0] = a$
 $\text{array}[\text{capacity} - 1] = b$
and all other entries are equally spaced values between a and b
 - It will return the address of the array



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

Lifetime of dynamically allocated memory 6

Allocating an array

- Allocate and initialize an array:

```
double *allocate( double a, double b, std::size_t const capacity ) {
    assert( capacity >= 2 );
    double *array{ new double[capacity] };

    // This will set array[k] = a + k*delta
    // so that array[0] = a
    // array[capacity - 1] = b
    double delta{ (b - a)/(capacity - 1.0) };

    for ( std::size_t k{0}; k < capacity; ++k ) {
        array[k] = a + k*delta;
    }

    return array;
}
```

3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 4.8 5.0



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

Lifetime of dynamically allocated memory 7

Manipulating that array:

- Let us let the user manipulate the array

```
void manipulate( double *array, std::size_t const capacity ) {
    while ( true ) {
        std::size_t k{};

        std::cout << "Which entry do you want to change? ";
        std::cin >> k;

        if ( k >= capacity ) {
            return;
        }

        std::cout << "What is the new value? ";
        std::cin >> array[k];
    }
}
```



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

Lifetime of dynamically allocated memory 8

Deallocating an array

- Clearing and deallocating the array

```
void deallocate( double *array, std::size_t const capacity ) {
    for ( std::size_t k{0}; k < capacity; ++k ) {
        array[k] = 0.0;
    }

    delete[] array;
    array = nullptr;
}
```



UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Lifetime of dynamically allocated memory

Using our functions

```
int main() {
    double *a_data{};

    std::size_t data_capacity{};
    std::cout << "Enter an array capacity: ";
    std::cin >> data_capacity;

    a_data = allocate( 0.0, 10.0, data_capacity );



    manipulate( a_data, data_capacity );

    std::cout << a_data[0];

    for ( std::size_t k{1}; k < data_capacity; ++k ) {
        std::cout << ", " << a_data[k];
    }

    deallocate( a_data, data_capacity );
    a_data = nullptr;

    return 0;
}
```

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGIES



Lifetime of dynamically allocated memory

Poor design?

- It seems a little awkward to have a function that clears and deallocates the array sent to it...
 - Perhaps we should just have a function


```
void clear( double *array, std::size_t const capacity );
```
 - Then our code would look like:


```
clear( a_data, data_capacity );
delete[] a_data;
a_data = nullptr;
```
 - These problems will be fixed later when we look at objects



UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Lifetime of dynamically allocated memory

Static versus dynamic memory

- If we call the `manipulate(...)` function with a local array
 - That is, one that is allocated on the call stack
 the function continues to work: it will manipulate that array
- If you call `deallocate(...)` with a local array, your program will crash: you cannot delete a local array
- You can try this yourself:


```
int main() {
    int array[100];
    delete[] array;
    return 0;
}
```

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Lifetime of dynamically allocated memory

Local variables



- It is important to differentiate between local variables and dynamically allocated memory


```
double *allocate( double a, double b, std::size_t const capacity ) {
    assert( capacity >= 2 );
    double *array{ new double[capacity] };

    double delta{ (b - a)/(capacity - 1.0) };

    for ( std::size_t k{0}; k < capacity; ++k ) {
        array[k] = a + k*delta;
    }

    return array;
}
```

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

Lifetime of dynamically allocated memory 13

Summary

- Following this lesson, you now
 - Have seen that we can pass or return dynamically allocated memory by passing or returning addresses (pointers)
 - Are aware it is important to make sure that you keep track of the address, otherwise you lose that memory
 - Understand that any function can call `new` or `delete` as long as the appropriate address is correctly dealt with
 - Know that when a local variable storing the address of dynamically allocated memory goes out of scope, it is only that local variable that is lost: the memory is still allocated



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

Lifetime of dynamically allocated memory 14

References

- [1] [https://en.wikipedia.org/wiki/New_and_delete_\(C++\)](https://en.wikipedia.org/wiki/New_and_delete_(C++))



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

Lifetime of dynamically allocated memory 15

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.



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

Lifetime of dynamically allocated memory 16

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.

