

Classes and Objects in C++ programming language

"Classes, Objects, Functions, Data Hiding"

Fundamentals of OOPs

Shakirullah Waseeb
shakir.waseeb@gmail.com

Nangarhar University

October 4, 2017



Agenda

- 1 Constructors and Destructors
 - Introduction
 - Constructor overloading
 - Default Copy Constructor
 - Destructors
- 2 Handling Objects in Functions
 - Passing objects as argument to a function
 - Returning objects from functions
- 3 What's next?
- 4 Questions and Discussion



Agenda

1 Constructors and Destructors

- Introduction
- Constructor overloading
- Default Copy Constructor
- Destructors

2 Handling Objects in Functions

- Passing objects as argument to a function
- Returning objects from functions

3 What's next?

4 Questions and Discussion



Introduction

- Sometimes it is convenient to initialize one or more member data upon creation of objects
- Consider the **Counter** example, which hold a counter variable, an increment and get_counter function; and can be used as a general purpose counter
- Initially upon creation of Counter instance variable the counter must be initialized with 0, so it would be convenient to have such a behavior
- This can be achieved using **constructor**; a **special member function** called automatically upon object instantiation



Syntax

- Constructor has same syntax with that of a function except; has no return type and its name is same with class name
- It must always be declared under public access specifier scope

counter.cpp

```
#include <iostream >
using namespace std;

class Counter {
private:
    int counter;
public:
    Counter () {
        counter = 0;
    }
    void increment() {
        counter += 1;
    }
    int get_counter() {
        return counter;
    }
};
```

Syntax - - - continue

- We can also use the **Initializer** list syntax to initialize data members with constructor as:

```
Counter() : counter(0) {}
```

```
SomeOtherClass() : Id(5), Price(2.4), Comment("Hi") {}
```



Agenda

1 Constructors and Destructors

- Introduction
- **Constructor overloading**
- Default Copy Constructor
- Destructors

2 Handling Objects in Functions

- Passing objects as argument to a function
- Returning objects from functions

3 What's next?

4 Questions and Discussion



Constructor Overloading

- When we don't define a no-argument constructor explicitly, an implicit no-argument constructor is built in to the program by compiler automatically
- This constructor is called *default constructor*, which enable us to creates the object
- We can overload constructor the way we overload the functions
- In case of overloaded constructors, while, creating the objects we must specify which constructor we wants to initialize the object

```
Counter(): counter(0) {}
```

```
Counter(int c): counter(c) {}
```

```
SomeOtherClass(int id, float price, string comment): Id(id),  
Price(price), Comment(comment) {}
```

Instantiation:

```
Counter counter1, counter2(4);
```



Agenda

1 Constructors and Destructors

- Introduction
- Constructor overloading
- **Default Copy Constructor**
- Destructors

2 Handling Objects in Functions

- Passing objects as argument to a function
- Returning objects from functions

3 What's next?

4 Questions and Discussion



Default Copy Constructor

- We studied two ways to initialize objects:
 - no-argument constructor:** initialize data members to constant values
 - multi-argument constructor:** initialize data members to values passed as arguments
- Another way to initialize an object! : *initialize it with another object of the same data type*
- Surprisingly, there exists a built into special constructor for all classes that do this for us:
 - default copy constructor:* a one argument constructor with an argument object of the same class as the constructor



Agenda

1 Constructors and Destructors

- Introduction
- Constructor overloading
- Default Copy Constructor
- Destructors

2 Handling Objects in Functions

- Passing objects as argument to a function
- Returning objects from functions

3 What's next?

4 Questions and Discussion



Destructors



Agenda

- 1 Constructors and Destructors
 - Introduction
 - Constructor overloading
 - Default Copy Constructor
 - Destructors
- 2 Handling Objects in Functions
 - Passing objects as argument to a function
 - Returning objects from functions
- 3 What's next?
- 4 Questions and Discussion





Agenda

- 1 Constructors and Destructors
 - Introduction
 - Constructor overloading
 - Default Copy Constructor
 - Destructors
- 2 Handling Objects in Functions
 - Passing objects as argument to a function
 - Returning objects from functions
- 3 What's next?
- 4 Questions and Discussion





What's next?

Inheritance

a) Derived and Base Classes, Derived Class Constructors



Your Turn: Time to hear from you!



1

¹<https://fensafitters.files.wordpress.com/2013/07/3d095.jpg>



References



Robert Lafore

Object-Oriented Programming in C++, 4th Edition .
2002.



Piyush Kumar

Object oriented Programming (Using C++)
<http://www.compgeom.com/piyush/teach/3330>

