## **Operator Overloading**

# "Overloading unary and binary operator" Fundamentals of OOPs

Shakirullah Waseeb shakir.waseeb@gmail.com

Nangarhar University

October 30, 2017





## Agenda

- Understanding Operator Overloading
  - Introduction
  - Syntax for operator overloading
- Overloading Unary Operator
- Overloading Binary Operator
- Questions and Discussion





## Agenda

- Understanding Operator Overloading
  - Introduction
  - Syntax for operator overloading
- Overloading Unary Operator
- Overloading Binary Operator
- Questions and Discussion





#### Introduction

- Operator Overloading: defining the behavior of c++ operators (+,\*,<=,>=, ++) when applied to user defined data types
   e.g. object3 = object1+object2
- It gives us the opportunity to redefine the C++ language by:
  - using classes to create new data types or new kind of variables
  - and operator overloading to create new definitions for operators
- Data Type Conversion, also connected with operator overloading;
  - conversion between simple data types in c++ is handled automatically
  - while conversions between user-defined data types required some work on programmer's side





## Agenda

- Understanding Operator Overloading
  - Introduction
  - Syntax for operator overloading
- Overloading Unary Operator
- Overloading Binary Operator
- Questions and Discussion





## Operator overloading syntax

• The syntax for overloading c++ operators is as follow:

```
return type operator operator_symbol (parameter_list){
  //implementation logic
}
```

Consider an example for class Counter

```
void operator ++ () {
    ++count;
```





## Overloading Unary Operator

- Unary operators: operators that act only on one operands ++,-, -
- In Counter example we had an increment example where the count member variable were increment by one each time the increment function is called
- What if we use ++objcounter; instead of objcounter.increment();

#### Counter Example

```
#include <iostream >
using namespace std;
class Counter {
    private:
        int count;
    public:
        Counter ():count(0) {}
        void operator ++ () {
            ++count;
        }
        void print () {
            cout « count«endl;
        }
};
```

## Overloading Unary Operator -Continue

#### Counter Example

```
int main() {
    Counter c1;
    ++c1;
    c1.print();
    ++c1;
    c1.print();
}
```





## Operator return values

- ullet We get a problem with previous implementation of the ++ operator in following situation
- Counter c2 = ++c1;

#### Counter Example

```
#include <iostream >
using namespace std;
class Counter {
    private:
        int count;
    public:
        Counter ():count(0) {}
        Counter operator ++ () {
            ++count;
            Counter temp;
            temp.count = count;
            return temp;
    }
    void print () {
            cout « count«endl;
        }
};
```

## Overloading Binary Operator



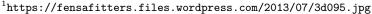


## Your Turn: Time to hear from you!



1





#### 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



