Object Oriented Programming Lab

Spring-2024

LAB 08

FAST National University of
Computer and Emerging Sciences

Learning Outcomes

In this lab you are expected to learn the following:

- Operator Overloading

Operator Overloading:

C++ allows you to specify more than one definition for an operator in the same scope, which is called operator overloading. Overloaded operators are functions with special names: the keyword "operator" followed by the symbol for the operator being defined.

Syntax for C++ Operator Overloading:

To overload an operator, we use a special **operator** function. We define the function inside the class or structure whose objects/variables we want the overloaded operator to work with.

```
class className {
    ... .. ...
    public
       returnType operator symbol (arguments) {
           ... .. ...
       }
    ... .. ...
};
```

Here,

- **returnType** is the return type of the function.
- **operator** is a keyword.
- **symbol** is the operator we want to overload. Like: +, <, -, ++, etc.
- **arguments** is the arguments passed to the function.

Lab Tasks

Submission Instructions:

1. Create a new folder with name *ROLLNO_SEC_LAB08* **e.g. i22XXXX_A_LAB08**
2. Move all of your .**cpp and .h files** to this newly created directory and compress it into a **.zip file**.
3. Now you have to submit this zipped file on Google Classroom.
4. If you don't follow the above-mentioned submission instruction, you will be marked **zero.**
5. Plagiarism in the Lab Task will result in **zero** marks in the whole category.

Q1. Create a class **Money** that represents a money value (combination of **dollars** and **cents**).

The class has the following member functions.

1. **Money()** // Default Constructor

   Initializes dollars and cents to zero

2. **Money(int dollar, int cents)** // Parameterized Constructor

   Update dollar and cents accordingly

Overload the following operators:

3. **Money operator+(const Money &obj)**

   Define an operator + that overloads the standard + math operator and allows one Money object to be added to another.

4. **Money operator-(const Money &obj)**

   Define an operator - that overloads the standard - math operator and allows one Money object to be added to another.

5. **const Money operator=(const Money &obj)**

   Define an operator = that overloads the standard = operator and assigns one Money object to be added to another.

6. **Money& operator++ ()**

   Overload pre increment operator.

7. **Money& operator-- ()**

   Overload pre decrement operator.

8. **bool operator!= (const Money& right)**

   Overload not equal operator that checks whether the right object is equal to the current object or not.

9. **void operator~ ()**

   Overload ~ operator to display dollars and cents in any money object.

Example:

1. M1: 12 dollars 95 cents using parameterized constructor
2. M2: 3 dollars 98 cents using parameterized constructor
3. Perform all the operations as shown below

Q2. Implementation of Quadratic Polynomial Class:

Write a class **Polynomial**. This class has three private data members:

a. int that holds the coefficient of X^2
b. int that holds the coefficient of X
c. A double that holds the coefficient of X^0 (Constant term)

The class has the following member functions.

Polynomial	Constructs a new Polynomial object to represent the quadratic Polynomial with all coefficients =0
Polynomial (a,b,c)	Constructs a new Polynomial object to represent the quadratic Polynomial
getters/setters	Write getter/setters for all members e.g. a,b,c
Polynomial(const & Copy)	Copy Constructor
operator =	Overload = operator to assign values
operator ==	Overload == comparison operator
Polynomial p3=p1+p2	Overload + operator which takes two Polynomial object as argument. It preforms the addition and returns the result.
Polynomial p3=p1-p2	Overload - operator which takes two Polynomial object as argument. It preforms the subtraction and returns the result.

Polynomial p2= p1*d	Overload * operator which takes an int as argument. It preforms the scaler multiplication and returns the result.

Example:

4. P1: 2x^2 +x -1, P2: 5x^2 -7x +3
5. Perform all the operations as shown below

```
P1:
2x^2 +1x -1

P2:
5x^2 -7x +3

P3:
7x^2 -6x +2

P4:
-3x^2 +8x -4

P5:
10x^2 +5x -5

P1 == P2
0
```