FAST School of Computing

Object Oriented Programming – Spring 2023

Cyber Security Department

LAB 09

Operator Overloading in C++

Learning Outcomes

In this lab you are expected to learn the following:

- Operator Overloading in Classes (C++)

Note: Plagiarism(from some else or internet) in any 1 question will lead to zero marks in the whole lab task.

What is Operator overloading?

In C++, it can add special features to the functionality and behavior of already existing operators like arithmetic and other operations. The mechanism of giving special meaning to an operator is known as operator overloading. For example, we can overload an operator '+' in a class-like string to concatenate two strings by just using +.

Restrictions on Operator Overloading

- C++ operators that can be overloaded

Operators that can be overloaded							
+	-	*	/	%	^	&	|
~	!	=	<	>	+=	-=	*=
/=	%=	^=	&=	|=	<<	>>	>>=
<<=	==	!=	<=	>=	&&	||	++
--	->*	,	->	[]	()	new	delete
new[]	delete[]						

- C++ Operators that cannot be overloaded

Operators that cannot be overloaded				
.	.*	::	?:	sizeof

Example

Suppose we have created three objects:

c1, c2 and **c3** from a class named **Complex** that represents complex numbers:

Till now if we want to add two complex number we do something like this:

```
result = c1.addNumbers(c2);
```

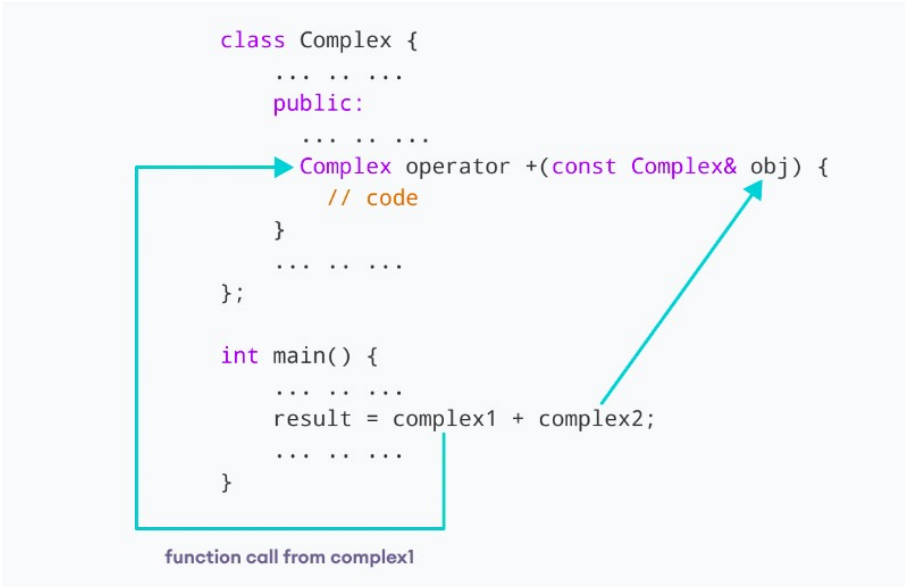
But, **operator overloading** allows us to change how operators work, we can redefine how the + operator works and use it to add the complex numbers of c1 and c2 by writing the following code:

```
result = c1 + c2;
```

But how will the compiler know that what does the + operator mean in this scenario?

ANSWER: You will have to define the functionality of + operator (operator overloading) for this class like this:

```
    // Overload the + operator
    Complex operator + (const Complex& obj) {
        Complex temp;
        temp.real = real + obj.real;
        temp.imag = imag + obj.imag;
        return temp;
    }
```



Problem 1:

Write a class **Distance** that holds distances or measurements expressed in feet and inches.

This class has **two private data members**:

feet: An integer that holds the feet.

inches: An integer that holds the inches.

1. Write a constructor with default parameters that initializes each data member of the class. If inches are greater than equal to 12 then they must be appropriately converted to corresponding feet
2. Generate appropriate **getter-setter** functions for the data members.

   bool setFeet(int f)

   int getFeet()const

   bool setInches(int i)

   It should ensure proper conversion to feet.

   int getInches()const
3. Define an **operator +** that overloads the standard + math operator and allows one Distance object to be added to another.

   Distance operator+(const Distance &obj)
4. Define an **operator-** function that overloads the standard - math operator and allows subtracting one Distance object from another.

   Distance operator-(const Distance &obj)
5. Define an **operator=** function that overloads the = operator and assign one Distance object to another.

   const Distance operator=(const Distance &obj)

Problem 2:

Write a class **Matrix.**

This class has **three private data members:**

rows: An integer that holds the numbers of rows for matrix

columns: An integer that holds the numbers of columns for matrix

matrix: An integer pointer to pointer that points to 2D array (rows x columns).

The class has the following member functions:

Matrix (int r, int c)	Constructs a new Matrix object to represent the given matrix
operator =	Overload = operator to assign values
operator ==	Overload == operator to compare whether matrices are equal or not
M2=M1+1	Overload + operator which takes integer as argument. It preforms scalar addition.
M2=M1-4	Overload - operator which takes integer as argument. It preforms scalar subtraction.
M3=M1+M2	Overload + operator which takes matrix object as argument. It adds two matrixes and returns the result.
M3=M1-M2	Overload - operator which takes matrix object as argument. It subtracts two matrixes and returns the result.

Submission Details:

1. Save single .cpp file with your roll no and lab number e.g. i22-XXXX_Lab9.cpp
2. Take screen shot of running test cases of tasks.
3. Zip the .cpp file and screen shots (Do not create .rar file) with roll no and lab no. e.g. i22-XXXX_Lab9.zip.
4. Submit the zip file on google class room.