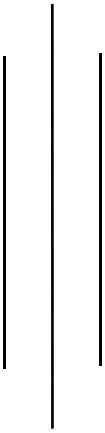
TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS



LAB REPORT ON

OBJECT ORIENTED PROGRAMMING [CT 451]

LAB 3

OPERATOR OVERLOADING

Submitted by:

Rujal Acharya

PUL076BEI029

Submitted to:

Department of Electronics and Computer Engineering, Pulchowk Campus

Institute of Engineering, Tribhuvan University

Lalitpur, Nepal

November, 2020

Task 1

Problem:

Write a program in CPP to overload unary ++ for postfix and prefix increment operation on complex numbers using member function.

Program:

```
#include <iostream>

class Complex {
    public:
    Complex();
    Complex(float, float);
    Complex operator ++();
    Complex operator ++(int);
    void getdata();
    void showdata();

    private:
    float re, imz;
};

Complex::Complex() {
    re = 0;
    imz = 0;
}

Complex::Complex(float r, float i) {
    re = r;
    imz = i;
}

Complex Complex::operator ++ () {
    Complex temp;
    temp.re = ++re;
    temp.imz = ++imz;
    return temp;
}

Complex Complex::operator ++ (int) {
    Complex temp;
    temp.re = re++;
```

```
        temp.imz = imz++;
        return temp;
}

void Complex::getdata() {
        std::cout << "Enter real and imaginary parts" << std::endl;
        std::cin >> re >> imz;
}

void Complex::showdata() {
        if (imz < 0) {
                std::cout << re << imz << "i" << std::endl;
        } else {
                std::cout << re << "+" << imz << "i" << std::endl;
        }
}

int main() {
        Complex a, b, c, d;
        a.getdata();
        b.getdata();
        c = a++;
        d = ++b;
        a.showdata();
        b.showdata();
        c.showdata();
        d.showdata();
        return 0;
}
```

Task 2

Problem:

Write a program in CPP to overload unary ++ for postfix and prefix increment operation on complex numbers using non-member/friend function.

Program:

```
#include <iostream>

class Complex {
    public:
    Complex();
    Complex(float, float);
    friend Complex operator ++(Complex &);
    friend Complex operator ++(Complex &, int);
    void getdata();
    void showdata();

    private:
    float re, imz;
};

Complex::Complex() {
    re = 0;
    imz = 0;
}

Complex::Complex(float r, float i) {
    re = r;
    imz = i;
}

Complex operator ++ (Complex &a) {
    Complex temp;
    temp.re = ++a.re;
    temp.imz = ++a.imz;
    return temp;
}

Complex operator ++ (Complex &a, int) {
    Complex temp;
    temp.re = a.re++;
```

```
        temp.imz = a.imz++;
        return temp;
}

void Complex::getdata() {
        std::cout << "Enter real and imaginary parts" << std::endl;
        std::cin >> re >> imz;
}

void Complex::showdata() {
        if (imz < 0) {
                std::cout << re << imz << "i" << std::endl;
        } else {
                std::cout << re << "+" << imz << "i" << std::endl;
        }
}

int main() {
        Complex a, b, c, d;
        a.getdata();
        b.getdata();
        c = a++;
        d = ++b;
        a.showdata();
        b.showdata();
        c.showdata();
        d.showdata();
        return 0;
}
```

Task 3

Problem:

Write a program in CPP to find the sum of two complex numbers using the concept of the overloading binary + operator using member function.

Program:

```
#include <iostream>

class Complex {
	public:
	Complex();
	Complex(float, float);
	void getdata();
	void showdata();
	Complex operator + (Complex);

	private:
	float re;
	float imz;
};

Complex::Complex() {
	re = 0;
	imz = 0;
}

Complex::Complex(float r, float i) {
	re = r;
	imz = i;
}

void Complex::getdata() {
	std::cout << "Enter real and imaginary parts" << std::endl;
	std::cin >> re >> imz;
}

void Complex::showdata() {
	if (imz < 0) {
		std::cout << re << imz << "i" << std::endl;
	} else {
		std::cout << re << "+" << imz << "i" << std::endl;
```

```
    }
}

Complex Complex::operator +(Complex a) {
    Complex temp;
    temp.re = re + a.re;
    temp.imz = imz + a.imz;
    return temp;
}

int main() {
    Complex a, b, sum;
    a.getdata();
    b.getdata();
    sum = a + b;
    sum.showdata();
    return 0;
}
```

Task 4

Problem:

Write a program in CPP to find the sum of two complex numbers using the concept of the overloading binary + operator using non-member/friend function.

Program:

```
#include <iostream>

class Complex {
	public:
	Complex();
	Complex(float, float);
	void getdata();
	void showdata();
	friend Complex operator + (Complex, Complex);

	private:
	float re;
	float imz;
};

Complex::Complex() {
	re = 0;
	imz = 0;
}

Complex::Complex(float r, float i) {
	re = r;
	imz = i;
}

void Complex::getdata() {
	std::cout << "Enter real and imaginary parts" << std::endl;
	std::cin >> re >> imz;
}

void Complex::showdata() {
	if (imz < 0) {
		std::cout << re << imz << "i" << std::endl;
	} else {
		std::cout << re << "+" << imz << "i" << std::endl;
```

```
    }
}

Complex operator +(Complex a, Complex b) {
    Complex temp;
    temp.re = a.re + b.re;
    temp.imz = a.imz + b.imz;
    return temp;
}

int main() {
    Complex a, b, sum;
    a.getdata();
    b.getdata();
    sum = a + b;
    sum.showdata();
    return 0;
}
```

Task 5

Problem:

Write a program in CPP to find the product of two 3 by 3 matrices entered by the user by overloading binary * operator.

Program:

```
#include <iostream>

class Matrix {
	public:
	void getdata();
	void showdata();
	Matrix operator * (Matrix);

	private:
	int mat[3][3];
};

void Matrix::getdata() {
	for (int i = 0; i < 3; i++) {
		for (int j = 0; j < 3; j++) {
			std::cout << "Enter value of row " << i+1 << ", column " << j+1 << " : ";
			std::cin >> mat[i][j];
		}
	}
}

void Matrix::showdata() {
	for (int i = 0; i < 3; i++) {
		for (int j = 0; j < 3; j++) {
			std::cout << mat[i][j] << "\t" ;
		}
	std::cout << std::endl;
	}
}

Matrix Matrix::operator * (Matrix A) {
	Matrix temp;
	for (int i = 0; i < 3; i++) {
		for (int j = 0; j < 3 ; j++) {
			int sum = 0;
```

```
                for (int k = 0; k < 3 ; k++) {
                    sum += mat[i][k] * A.mat[k][j];
                }
                temp.mat[i][j] = sum;
            }
        }
        return temp;
}

int main() {
        Matrix A, B, SUM;
        A.getdata();
        B.getdata();
        SUM = A * B;
        SUM.showdata();
        return 0;
}
```