**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **ПЗ**



### ЗВІТ

До лабораторної роботи № 6

**На тему:** *“* *Перевантаження функцій і операцій, дружні функції”*

**З дисципліни:** *“Об’єктно-орієнтоване програмування”*

**Лектор:**

доцент каф.ПЗ

Коротєєва Т. О.

**Виконав:**

ст. гр. ПЗ-16

Шеремета А.І.

**Прийняв:**

асист. каф. ПЗ

Дивак І.В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_ .

Львів – 2022

**Тема роботи:** Перевантаження функцій і операцій, дружні функції.

**Мета** **роботи:** Навчитися використовувати механізм перевантаження функцій та операцій. Навчитися створювати та використовувати дружні функції. Ознайомитися з статичними полями та методами, а також навчитися їх використовувати при створенні класів.

**Теоретичні відомості**

Перевантаження функції, перевантаження процедури або ж перевантаження методу це в програмуванні один із засобів реалізації поліморфізму, що полягає в можливості створювати кілька реалізацій функції із тим же ім'ям проте з різною кількістю параметрів або з різним типом параметрів.

Перевантаження операторів це один із засобів реалізації поліморфізму що полягає в можливості одночасного існування в одній зоні видимості декількох різних варіантів застосування операторів, що мають одне й те саме ім'я, але різні типи аргументів, до яких вони застосовуються.

Дружня функція – це функція, яка не є членом класу, але має доступ до членів класу, оголошеним в полях private або protected. У всіх інших відносинах дружня функція є звичайною функцією. Нею може бути, як проста функція, так і метод іншого класу.

**Варіант 2**

Клас Complex – комплексне число.

Перевантажити операції, як функції члени:

Додавання

Віднімання

Множення

Піднесення до n-го степеня

Перевантажити операції, як дружні-функції:

Введення комплексного числа з форми ("<<")

Виведення комплексного числа на форму(">>")

Більше (">")

Менше ("<")

Рівне ("==") (при порівнянні порівнювати модулі комплексних чисел).

Створити статичне поле, в якому б містилась інформація про кількість створених об’єктів, а також статичні функції для роботи з цим полем.

**Лабораторне завдання**

1. Створити клас відповідно до варіанту (див. Додаток).
2. Перевантажити як мінімум три функції-члени з попереднього завдання.
3. Перевантажити операції згідно з варіантом (див. Додаток). Для операцій, для яких не вказані символи, вибрати символи самостійно.
4. Створити дружні функції згідно з варіантом.
5. Створити статичні поля та статичні методи згідно з варіантом.
6. Продемонструвати розроблені можливості класу завдяки створеному віконному застосуванню.
7. Оформити звіт до лабораторної роботи.

**Хід роботи**

1. Реалізовую програму:

**Файл mainwindow.cpp**

int complex::objCount = 0;

complex b,c,e,f,g,d,s;

complex complex::operator+(complex const &c)

{

R1 +=c.R1;

Im1 +=c.Im1;

R2 +=c.R2;

Im2 +=c.Im2;

return \*this;

}

complex complex::operator=(complex const &c)

{

R1 = c.R1;

Im1 = c.Im1;

R2 = c.R2;

Im2 = c.Im2;

return \*this;

}

complex complex::operator-(complex const &c)

{

R1 -= c.R1;

Im1 -= c.Im1;

R2 -= c.R2;

Im2 -= c.Im2;

return \*this;

}

double complex::operator\*(const complex& c){

return R1\*c.R1 + Im1\*c.Im1 + R2\*c.R2 + Im2\*c.Im2;

}

complex operator\*(complex &c, double a){

c.R1\*=a;

c.Im1\*=a;

c.R2\*=a;

c.Im2\*=a;

return c;

}

bool operator>(const complex A, const complex B){

return sqrt(A.R1 \* A.R1 + A.Im1 \* A.Im1) > sqrt(B.R1 \* B.R1 + B.Im1 \* B.Im1);

}

bool operator<(const complex A, const complex B){

return sqrt(A.R1 \* A.R1 + A.Im1 \* A.Im1) > sqrt(B.R1 \* B.R1 + B.Im1 \* B.Im1);

}

void operator>>(complex &c, QTableWidget \*tableWidget) {

c.setR1(tableWidget->item(0,0)->text().toDouble());

c.setIm1(tableWidget->item(0,1)->text().toDouble());

c.setR2(tableWidget->item(0,2)->text().toDouble());

c.setIm2(tableWidget->item(0,3)->text().toDouble());

}

void operator<<(complex &c, QTableWidget \*tableWidget) {

tableWidget->item(0,0)->setText(QString::number(c.getR1()));

tableWidget->item(0,1)->setText(QString::number(c.getIm1()));

tableWidget->item(0,2)->setText(QString::number(c.getR2()));

tableWidget->item(0,3)->setText(QString::number(c.getIm2()));

}

void complex::setR1(double R1\_) {R1 = R1\_; };

void complex::setIm1(double Im1\_) {Im1 = Im1\_; };

void complex::setR2(double R2\_) {R1 = R2\_; };

void complex::setIm2(double Im2\_) {Im2 = Im2\_; };

double complex::getR1() const { return R1; };

double complex::getIm1() const { return Im1; };

double complex::getR2() const { return R2; };

double complex::getIm2() const { return Im2; };

**Файл mainwindow.h**

class complex

{

public:

complex(){objCount++;};

complex (const complex& c){R1 = c.R1; Im1 = c.Im1; R2 =c.R2; Im2 = c.Im2;objCount++;};

complex(double r1, double im1, double r2, double im2){

R1 = r1; Im1 = im1; R2 = r2; Im2 = im2; objCount++;

};

~complex() {objCount--;};

public:

complex operator=(const complex &с);

complex operator+(const complex &с);

complex operator-(const complex &с);

complex operator^(const complex &с);

double operator\*(const complex &с);

friend complex operator\*(complex &c, double a);

friend void operator>>(complex &c, QTableWidget \*tableWidget);

friend void operator<<(complex &c, QTableWidget \*tableWidget);

friend bool operator>(const complex c, const complex с2);

friend bool operator<(const complex c, const complex с2);

friend bool operator==(const complex c, const complex с2);

static int objCount;

static int getObjCount(){

return objCount;}

double sumi();

double sumr();

double subtractioni();

double subtractionr();

double multiplyi();

double multiplyr();

double divisioni();

double divisionr();

double npowerr(int n);

double npoweri(int n);

double nrootr(int n);

double nrooti(int n);

void setR1(double R1\_);

void setIm1(double Im1\_);

void setR2(double R2\_);

void setIm2(double Im2\_);

double getR1() const;

double getIm1() const;

double getR2() const;

double getIm2() const;

void setnumbers(double Real, double Imaginary, double Real2, double Imaginary2)

{

R1 = Real;

Im1 = Imaginary;

R2 = Real2;

Im2 = Imaginary2;

};

private:

double R1,Im1,R2,Im2;

};

**Виконання лаборатоної роботи**

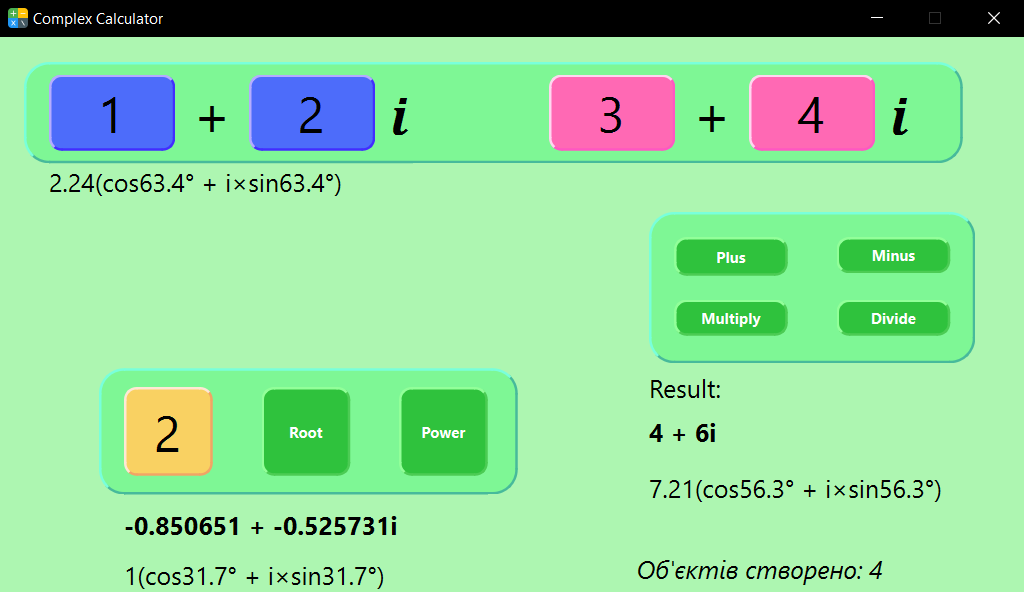


Рис. 1 Виконання програми

**Висновки**

На цій лабораторній роботі я навчився створювати класи, конструктори та функції-члени. Також я навчився перевантажувати функції та операції, створювати та використовувати дружні функції.