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.

Виконала: ст.гр. ПЗ-12

Кохман О.В.

Прийняла: доцент каф. ПЗ

Дяконюк Л.М.

«_____»_____2022р.

∑ ____ _____.

Львів – 2022

Тема: Перевантаження функцій і операцій, дружні функції, статичні методи класу.

Мета: Навчитися використовувати механізм перевантаження функцій та операцій. Навчитися створювати та використовувати дружні функції. Ознайомитися зі статичними полями і методами та навчитися їх використовувати.

Теоретичні відомості

Перевизначення операцій.

C++ підтримує спеціальні засоби, які дозволяють перевизначити вже існуючі операції. Наприклад, для операції + можна ввести своє власне визначення, яке реалізує операцію додавання для об'єктів певного класу. Фактично пере визначення для операцій існує і в мові С. Так, операція + може використовувати як об'єкти типу int, так і об'єкти типу float. C++ розширює цю ідею.

Для визначення операції використовується функція, що вводиться користувачем. Тип функції визначається іменем класу, далі записується ключове слово operator, за яким слідує символ операції, в круглих дужках дається перелік параметрів, серед яких хоча б один типу клас.

Функція **operator+** повертає результат типу **complex** та має параметри C1 та C2 типу **complex.** Функції-операції мають бути нестатичними функціями-членами класу або мати мінімум один аргумент типу класу. За виключенням операції присвоєння всі перевизначені оператори наслідуються.

Дружні функції

Дружньою функцією класу називається функція, яка сама не є членом класу, але має повні права на доступ до закритих та захищених елементів класу. Оскільки така функція не є членом класу, то вона не може бути вибрана з допомогою операторів (.) та (->), Дружня функція класу викликається звичайним способом. Для опису дружньої функції використовується ключове слово **friend**.

Статичні змінні класу

До тепер рахувалось, що дані в кожному об’єкті є власністю саме цього об’єкту та не використовуються іншими об’єктами цього класу. Та іноді приходиться слідкувати за накопиченням даних. Наприклад, необхідно з’ясувати скільки об’єктів даного класу було створено на даний момент та скільки з них існує. Статичні змінні-члени досяжні для всіх екземплярів класу. Це є компроміс між глобальними даними, які досяжні всім елементам програми, та даними, що досяжні тільки об’єктам даного класу.

Статична змінні створюється в одному екземплярі для всіх об’єктів даного класу. Розглянемо приклад. Об’єкт класу Cat включає статичну змінну-член HowManyCats (скільки котів). Ця змінна нараховує кількість об’єктів класу Cat, що створені під час виконання програми. Для цього статична змінна HowManyCats збільшується на 1 при кожному виклику конструктора класу Cat , а при виклику деструктора зменшується на 1.

Статичні змінні необхідно обов’язково ініціалізувати. Якщо необхідно обмежити доступ до статичних змінних, то оголошуйте їх закритими або захищеними.

Статичні функції класу

Статичні функції класу подібні до статичних змінних: вони не належать одному об’єкту, а знаходяться в області дії всього класу. Статичні функції-члени не мають вказівника this. Відповідно їх не можна оголосити як const. Статичні функції-члени не можуть звертатись до нестатичних змінних. До статичних функцій-членів можна звертатись з об’єкту їх класу, або вказавши повне ім’я, включаючи ім’я об’єкту.

Індивідуальне завдання

На основі класу з попередньої лабораторної:

Перевантажити як мінімум три функції-члени з попереднього завдання.

Перевантажити операції згідно з варіантом (див. Додаток). Для операцій, для яких не вказані символи, вибрати символи самостійно.

Створити дружні функції згідно з варіантом.

Створити статичні поля та статичні методи згідно з варіантом.

Продемонструвати розроблені можливості класу завдяки створеному віконному застосуванню.

Оформити звіт до лабораторної роботи.

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.

Код програми

Назва файлу: MyForm.h

```
#pragma once
#include "Complex.h"
namespace ContinueComplex {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
```

```
		MyForm(void)
		{
			InitializeComponent();
		}

	protected:
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}

	protected:
	private: System::Windows::Forms::Label^ label8;
	private: System::Windows::Forms::Label^ label7;
	private: System::Windows::Forms::TextBox^ textBox12;
	private: System::Windows::Forms::TextBox^ textBox11;
	private: System::Windows::Forms::TextBox^ textBox10;
	private: System::Windows::Forms::TextBox^ textBox7;
	private: System::Windows::Forms::TextBox^ textBox6;
	private: System::Windows::Forms::Button^ button4;
	private: System::Windows::Forms::Button^ button3;
	private: System::Windows::Forms::Button^ button2;
	private: System::Windows::Forms::TextBox^ textBox5;
	private: System::Windows::Forms::Button^ button1;
	private: System::Windows::Forms::Label^ label4;
	private: System::Windows::Forms::Label^ label3;
	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::Button^ button5;
	private: System::Windows::Forms::Button^ button6;
	private: System::Windows::Forms::Button^ button7;
	private: System::Windows::Forms::TextBox^ textBox8;
	private: System::Windows::Forms::TextBox^ textBox9;
	private: System::Windows::Forms::TextBox^ textBox13;
	private: System::Windows::Forms::Button^ button8;
	private: System::Windows::Forms::TextBox^ textBox14;
	private: System::Windows::Forms::TextBox^ textBox1;
	private: System::Windows::Forms::TextBox^ textBox2;
	private:
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

#pragma endregion
private: System::Void mathOperations(System::Object^ sender, System::EventArgs^
e) {
	Complex result, first, second;
	textBox1 >> first;
	textBox2 >> second;
	System::Windows::Forms::Button^ button = (Button^)sender;
	System::String^ textOfButtons = button->Text;
		if (textOfButtons == "To Add") {
			result = first + second;
			textBox5 << result;
		}
		else if (textOfButtons == "To Substract") {
			result = first - second;
			textBox6 << result;
		}
		else if (textOfButtons == "To Multiply") {
			result = first * second;
```

```
			textBox7 << result;
		}
		else if (textOfButtons == "To Elevate") {
			result = first ^ System::Convert::ToInt16(textBox10->Text);
			textBox11 << result;
			result = second ^ System::Convert::ToInt16(textBox10->Text);
			textBox12 << result;
		}
		else if (textOfButtons == "==") {
		    textBox8->Text = System::Convert::ToString(first == second);
		}
		else  if (textOfButtons == ">") {
			textBox9->Text = System::Convert::ToString(first > second);
		}
		else  if (textOfButtons == "<") {
			textBox13->Text = System::Convert::ToString(first < second);
		}
		else if (textOfButtons == "static") {
			Complex::printStatic(textBox14);
		}
}
private: System::Void MyForm_Load(System::Object^ sender, System::EventArgs^ e) {
}
};
}
```

Назва файлу: MyForm.cpp

```
#include "MyForm.h"
using namespace ContinueComplex;
int main() {
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Application::Run(gcnew MyForm());
	return 0;
}
```

Назва файлу: Complex.h

```
#ifndef _COMPLEX_H
#define _COMPLEX_H
#pragma once
#include <string>
#include <array>
#include <cmath>
float findPhi(float real, float imagine);
static std::string toStandardString(System::String^ string);
const double pi = 3.14159265358979323846;
class Complex {
private:
	float real;
	float imagine;
	static int numberOfObjects;
public:
	Complex();
	Complex(float _imagine);
	Complex(float _real, float _imagine);
	void setReal(float _real);
	void setImagine(float _imagine);
	float getReal() const;
	float getImagine() const;
	static float getStatic();
	static void printStatic(System::Windows::Forms::TextBox^ textBox);
```

```
    Complex operator+(const Complex& other) const;
      Complex operator-(const Complex& other) const;
      Complex operator*(const Complex& other) const;
      Complex operator^(const int n) const;
      friend bool operator==(const Complex& first, const Complex& second);
      friend bool operator>(const Complex& first, const Complex& second);
      friend bool operator<(const Complex& first, const Complex& second);
      friend void operator<<(System::Windows::Forms::TextBox^ textBox, const
Complex& other);
      friend void operator>>(System::Windows::Forms::TextBox^ textBox, Complex&
other);
      Complex addComplex(const Complex& other) const;
      Complex substractComplex(const Complex& other) const;
      Complex multiplyComplex(const Complex& other) const;
      Complex addComplex(const double number) const;
      Complex substactComplex(const double number) const;
      Complex multiplyComplex(const double number) const;
};
#endif
```

Назва файлу:Complex.cpp

```
#include "Complex.h"
Complex::Complex() : real(0), imagine(0) {
};
Complex::Complex(float _imagine) :imagine(_imagine), real(0) {
};
Complex::Complex(float _real, float _imagine) :real(_real), imagine(_imagine) {
      numberOfObjects++;
};
int Complex::numberOfObjects = 0;
void Complex::setReal(float _real) {
      real = _real;
}
void Complex::setImagine(float _imagine) {
      imagine = _imagine;
}
float Complex::getReal()const {
      return real;
}
float Complex::getImagine()const {
      return imagine;
}
float Complex::getStatic() {
      return numberOfObjects;
}
void Complex::printStatic(System::Windows::Forms::TextBox^ textBox) {
      textBox->Text = System::Convert::ToString(getStatic());
}
Complex Complex::operator+(const Complex& other) const {
      return Complex(real + other.real, imagine + other.imagine);
}
Complex Complex::operator-(const Complex& other) const {
      return Complex(real - other.real, imagine - other.imagine);
}
Complex Complex::operator*(const Complex& other) const {
      return Complex(real * other.real - imagine * other.imagine, real *
other.imagine + other.real * imagine);
}
Complex Complex::operator^(const int n) const {
      float ro = sqrt(real * real + imagine * imagine);
      float phi = findPhi(real, imagine);
      return Complex(pow(ro, n) * cos(n * phi), pow(ro, n) * sin(n * phi));
```

```
}
bool operator==(const Complex& first, const Complex& second) {
	return (sqrt(first.real * first.real + first.imagine * first.imagine) ==
sqrt(second.real * second.real + second.imagine * second.imagine)) ? true :
false;
}
bool operator<(const Complex& first, const Complex& second) {
	return (sqrt(first.real * first.real + first.imagine * first.imagine) <
sqrt(second.real * second.real + second.imagine * second.imagine)) ? true :
false;
}
bool operator>(const Complex& first, const Complex& second) {
	return (sqrt(first.real * first.real + first.imagine * first.imagine) >
sqrt(second.real * second.real + second.imagine * second.imagine)) ? true :
false;
}
void operator<<(System::Windows::Forms::TextBox^ textBox, const Complex& other) {
	System::String^ sign = (other.imagine >= 0) ? "+" : "-";
	textBox->Text = textBox->Text + System::Convert::ToString(other.real +
sign + fabs(other.imagine) + "i" + Environment::NewLine);
}
static std::string toStandardString(System::String^ string) {
	using System::Runtime::InteropServices::Marshal;
	System::IntPtr pointer = Marshal::StringToHGlobalAnsi(string);
	char* charPointer = reinterpret_cast<char*>(pointer.ToPointer());
	std::string returnString(charPointer, string->Length);
	Marshal::FreeHGlobal(pointer);
	return returnString;
}
void operator>>(System::Windows::Forms::TextBox^ textBox, Complex& other) {
	System::String^ soString = textBox->Text;
	std::string someString = toStandardString(textBox->Text);
	char str[10];
	strcpy(str, someString.c_str());
	char* token = strtok(str, " ");
	std::size_t offset = 0;
	other.real = std::stof(token, &offset);
	int i = 1;
	while (token != NULL) {
		token = strtok(NULL, " ");
		i++;
		if (i == 3) {
			other.imagine = std::stof(token,&offset);
		}
	}
}
Complex Complex::addComplex(const Complex& other) const {
	return Complex(real + other.real, imagine + other.imagine);
}
Complex Complex::substractComplex(const Complex& other) const {
	return Complex(real - other.real, imagine - other.imagine);
}
Complex Complex::multiplyComplex(const Complex& other) const {
	return Complex(real * other.real - imagine * other.imagine, real *
other.imagine + other.real * imagine);
}
Complex Complex::addComplex(const double number) const {
	return Complex(real + number, imagine);
}
Complex Complex::substactComplex(const double number) const{
	return Complex(real - number, imagine);
}
Complex Complex::multiplyComplex(const double number) const {
```

```
        return Complex(real * number, imagine * number);
}
float findPhi(float real, float imagine) {
        float phi = 0;
        if (real > 0 && imagine >= 0) {
                phi = atan(fabs(imagine / real));
        }
        else if (real < 0 && imagine >= 0) {
                phi = pi - atan(fabs(imagine / real));
        }
        else if (real < 0 && imagine < 0) {
                phi = pi + atan(fabs(imagine / real));
        }
        else if (real > 0 && imagine < 0) {
                phi = 2 * pi - atan(fabs(imagine / real));
        }
        else if (real == 0 && imagine > 0) {
                phi = pi / 2;
        }
        else if (real == 0 && imagine < 0) {
                phi = 3 * pi / 2;
        }
        return phi;
}
```

Протокол роботи

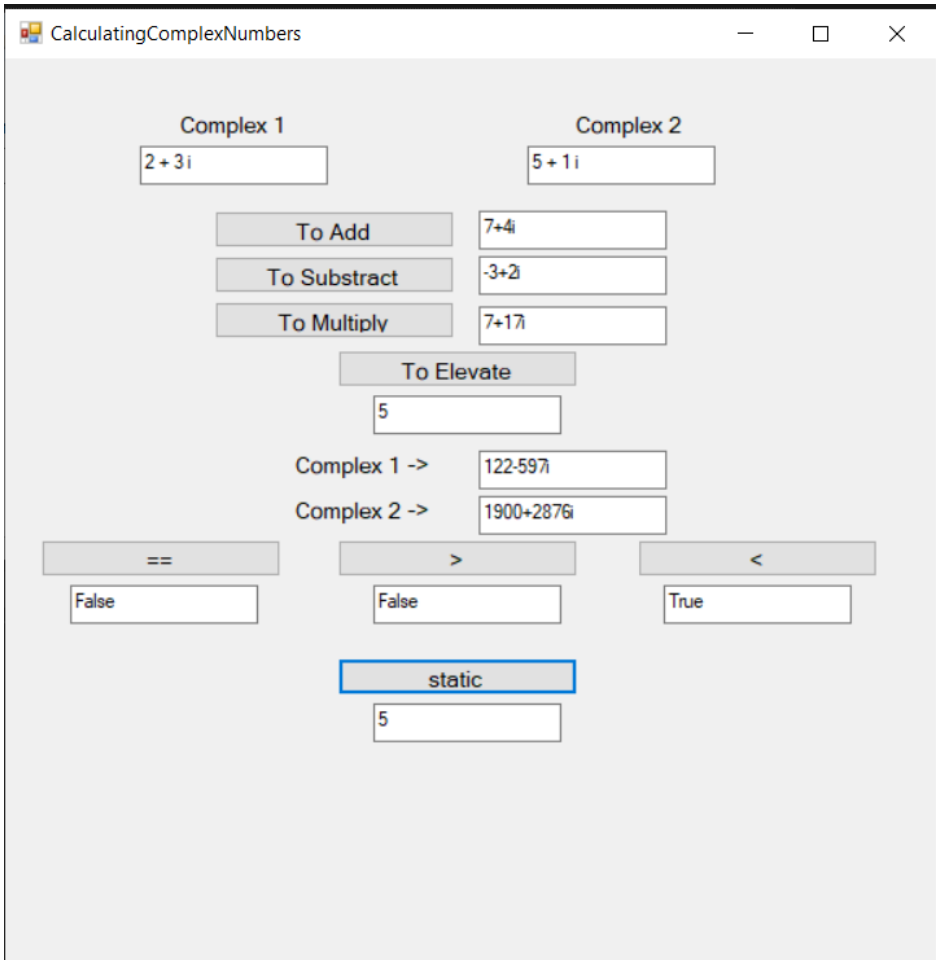

Рис. 1 Результат роботи програми.

Висновок

На цій лабораторній роботі я дізналась про дружні функції, про перевантаження функцій і операцій , про статичні змінні і методи класу, а також навчилась використовувати це все у своїй програмі і продемонструвала результати роботи за допомогою форми в Visual Studio 2022.