МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 6
На тему: "*Перевантаження функцій і операцій, дружні функції, статичні члени класу*"
З дисципліни: "Об'єктно-орієнтоване програмування"

Лектор:
доцент кафедри ПЗ
Коротєєва Т.О.

Виконав:
студент групи ПЗ-16
Коваленко Д.М.

Прийняв:
доцент кафедри ПЗ
Яцишин С.І.

«_____» _____ 2022 р.
$\sum$ = _____.....

Львів — 2022

Тема. Перевантаження функцій і операцій, дружні функції, статичні члени 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.

Лабораторне завдання

1. Створити клас відповідно до варіанту;
2. При створенні класу повинен бути дотриманий принцип інкапсуляції;
3. Створити конструктор за замовчуванням та хоча б два інших конструктори для початкової ініціалізації об'єкта;
4. Створити функції члени згідно з варіантом;
5. Продемонструвати можливості класу завдяки створеному віконному застосуванню;
6. У звіті до лабораторної намалювати UML-діаграму класу, яка відповідає варіанту.

Клас Triangle – трикутник на площині (задаються довжини трьох сторін).
Перевантажити операції, як функції члени:
Збільшення одразу всіх трьох сторін трикутника на константу("+")
Збільшення одразу всіх трьох сторін трикутника у певну кількість разів("*")
Доступ до і-ї сторони трикутника ("[ ]")
Перевантажити оператор приведення трикутника до дійсного типу(повертати його периметр).
Перевантажити операції, як дружні-функції:
Введення трикутника з форми ("<<")
Виведення трикутника на форму (">>")
Більше (">")
Менше ("<")
Рівне ("==") (при порівнянні порівнювати периметри).
Створити статичне поле, в якому б містилась інформація про кількість створених об'єктів, а також статичні функції для роботи з цим полем.

Теоретичні відомості

++ 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 C. 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.

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.

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.

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.

Код програми № 1

Назва файлу: *main.cpp*

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Назва файлу: *mainwindow.cpp*

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "triangle.h"

Triangle * t1 = new Triangle(1,1,1);
Triangle * t2 = new Triangle(1,1,1);

MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow) {
    ui->setupUi(this);
    ui->objectCountLabel->setText(QString::number(Triangle::getObjectCount()));
}

MainWindow::~MainWindow() {
    delete ui;
}

void MainWindow::on_set1Button_clicked() {
    Triangle * triangle = new Triangle();
    ui->objectCountLabel->setText(QString::number(Triangle::getObjectCount()));
    ui >> *triangle;
    delete t1;
    t1 = triangle;
}

void MainWindow::on_set2Button_clicked() {
    Triangle * triangle = new Triangle();
    ui->objectCountLabel->setText(QString::number(Triangle::getObjectCount()));
    triangle->setA(ui->a2LineEdit->text().toDouble());
    triangle->setB(ui->b2LineEdit->text().toDouble());
    triangle->setC(ui->c2LineEdit->text().toDouble());
    delete t2;
    t2 = triangle;
}

void MainWindow::on_areaPushButton_clicked() {
    ui->areaResultLabel->setText(QString::number((*t1).area()));
}

void MainWindow::on_anglesPushButton_clicked() {
    ui->anglesResultLabel->setText(QString::fromStdString(" a: ") + QString::
    number((*t1).angles(0)) +
    QString::fromStdString(" b: ") + QString::number((*t1).angles(1)) +
```

```
    QString::fromStdString(" c: ") + QString::number((*t1).angles(2)));
}

void MainWindow::on_perimeterPushButton_clicked() {
    (*t1).medians(0);
    ui->perimeterResultLabel->setText(QString::number((double)(*t1)));
}

void MainWindow::on_mediansPushButton_clicked() {
    ui->mediansResultLabel->setText(QString::fromStdString(" a: ") + QString::
    number((*t1).medians(0)) +
    QString::fromStdString(" b: ") + QString::number((*t1).medians(1)) +
    QString::fromStdString(" c: ") + QString::number((*t1).medians(2)));
}

void MainWindow::on_isRightPushButton_clicked() {
    ui->isRightLabel->setText((*t1).isRight() ? QString::fromStdString("true") :
     QString::fromStdString("false"));
}

void MainWindow::on_increasePushButton_clicked() {
    (*t1) += ui->increaseLineEdit->text().toDouble();
    ui << (*t1);
}

void MainWindow::on_increaseMPushButton_clicked() {
    (*t1) *= ui->increaseMLineEdit->text().toDouble();
    ui << (*t1);
}

void MainWindow::on_comparePushButton_clicked() {
    if ((*t1) > (*t2)) ui->compareResultLabel->setText(QString::fromStdString("
    Triangle 1 is bigger."));
    if ((*t2) > (*t1)) ui->compareResultLabel->setText(QString::fromStdString("
    Triangle 2 is bigger."));
    if ((*t1) == (*t2)) ui->compareResultLabel->setText(QString::fromStdString("
    Triangle 1 and 2 are equal."));
}
```

Назва файлу: *mainwindow.h*

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

    public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

    private slots:
    void on_set1Button_clicked();
    void on_set2Button_clicked();
    void on_areaPushButton_clicked();
```

```
    void on_anglesPushButton_clicked();
    void on_perimeterPushButton_clicked();
    void on_mediansPushButton_clicked();
    void on_isRightPushButton_clicked();
    void on_increasePushButton_clicked();
    void on_increaseMPushButton_clicked();
    void on_comparePushButton_clicked();

    private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

Назва файлу: *triangle.cpp*

```
#include "triangle.h"
#include <math.h>

int Triangle::objectCount = 0;

double Triangle::area() {
    double p = 0.5 * (a + b + c);
    double S = sqrt(p*(p-a)*(p-b)*(p-c));
    return S;
}

Values Triangle::angles() {
    Values angles;
    angles.a = acos((a*a - b*b - c*c) / ((-2)*b*c)) * 180/M_PI;
    angles.b = acos((b*b - a*a - c*c) / ((-2)*a*c)) * 180/M_PI;
    angles.c = acos((c*c - a*a - b*b) / ((-2)*a*b)) * 180/M_PI;
    return angles;
}

double Triangle::angles(int n) {
    double angle;
    if (n == 0) angle = acos((a*a - b*b - c*c) / ((-2)*b*c)) * 180/M_PI;
    else if (n == 1) angle = acos((b*b - a*a - c*c) / ((-2)*a*c)) * 180/M_PI;
    else if (n == 2) angle = acos((c*c - a*a - b*b) / ((-2)*a*b)) * 180/M_PI;
    else return 0.f;
    return angle;
}

double Triangle::perimeter() {
    return (double)*this;
}

Values Triangle::medians() {
    Values medians;
    medians.a = sqrt((2*b*b + 2*c*c - a*a) / 4);
    medians.b = sqrt((2*a*a + 2*c*c - b*b) / 4);
    medians.c = sqrt((2*a*a + 2*b*b - c*c) / 4);
    return medians;
}

double Triangle::medians(int n) {
    double median;
    if (n == 0) median = sqrt((2*b*b + 2*c*c - a*a) / 4);
    else if (n == 1) median = sqrt((2*a*a + 2*c*c - b*b) / 4);
    else if (n == 2) median = sqrt((2*a*a + 2*b*b - c*c) / 4);
    else return 0.f;
    return median;
```

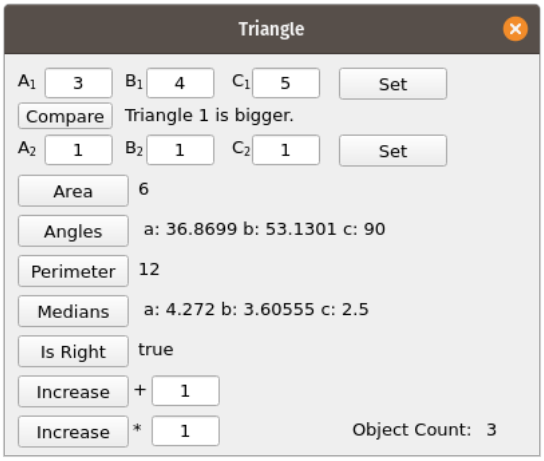
```
}

bool Triangle::isRight() {
    return angles(0) == 90.f || angles(1) == 90.f || angles(2) == 90.f;
}

void Triangle::increaseSides(double n) {
    *this += n;
}

void Triangle::operator+=(double n) {
    a += n;
    b += n;
    c += n;
}

void Triangle::operator*=(double n) {
    a *= n;
    b *= n;
    c *= n;
}

double Triangle::operator[](int i) {
    if (i==0) return a;
    else if (i==1) return b;
    else if (i==2) return c;
    else return 0;
}

Triangle::operator double() {
    return a + b + c;
}

void operator<<(Ui::MainWindow *ui, Triangle &t) {
    ui->a1LineEdit->setText(QString::number(t[0]));
    ui->b1LineEdit->setText(QString::number(t[1]));
    ui->c1LineEdit->setText(QString::number(t[2]));
}

void operator>>(Ui::MainWindow *ui, Triangle &t) {
    t.setA(ui->a1LineEdit->text().toDouble());
    t.setB(ui->b1LineEdit->text().toDouble());
    t.setC(ui->c1LineEdit->text().toDouble());
}

bool operator>(Triangle &t1, Triangle &t2) {
    return (double)t1 > (double)t2;
}

bool operator<(Triangle &t1, Triangle &t2) {
    return (double)t1 < (double)t2;
}

bool operator==(Triangle &t1, Triangle &t2) {
    return (double)t1 == (double)t2;
}
```

Назва файлу: *triangle.h*

```
#ifndef TRIANGLE_H
#define TRIANGLE_H
```

```
#include "ui_mainwindow.h"

struct Values {
    double a;
    double b;
    double c;
};

class Triangle {
    private:
    double a;
    double b;
    double c;
    static int objectCount;
    public:
    Triangle():
    Triangle(1.f, 1.f, 1.f)
    {};
    Triangle(double a, double b):
    Triangle(a, b, 1.f)
    {};
    Triangle(double a, double b, double c) {
        this->a = a;
        this->b = b;
        this->c = c;
        objectCount++;
    };

    double area();
    Values angles();
    double angles(int n);
    double perimeter();
    Values medians();
    double medians(int n);
    bool isRight();
    void increaseSides(double n);

    double getA() const { return a; };
    double getB() const { return b; };
    double getC() const { return c; };
    static int getObjectCount() { return objectCount; };
    void setA(double a) { this->a = a; };
    void setB(double b) { this->b = b; };
    void setC(double c) { this->c = c; };

    void operator+=(double n);
    void operator*=(double n);
    double operator[](int i);
    operator double();
    friend void operator<<( Ui::MainWindow *ui, Triangle &t );
    friend void operator>>(Ui::MainWindow *ui, Triangle &t);
    friend bool operator>(Triangle &t1, Triangle &t2);
    friend bool operator<(Triangle &t1, Triangle &t2);
    friend bool operator==(Triangle &t1, Triangle &t2);
};

#endif // TRIANGLE_H
```

Робота програми



UML діаграма

```
,-------------------------------------------------------------------------------.
| Triangle                                                                      |
|-------------------------------------------------------------------------------|
|- a : double                                                                   |
|- b : double                                                                   |
|- c : double                                                                   |
|- objectCount : int                                                            |
|+ Triangle() : Triangle                                                        |
|+ Triangle(a: double, b: double) : Triangle                                    |
|+ Triangle(a: double, b: double, c: double) : Triangle                         |
|+ area() : double                                                              |
|+ angles() : Values                                                            |
|+ angles(n : int) : double                                                     |
|+ perimeter() : double                                                         |
|+ medians() : Values                                                           |
|+ medians(n : int) : double                                                    |
|+ isRight() : bool                                                             |
|+ increaseSides(n : double)                                                    |
|+ getA() : double                                                              |
|+ getB() : double                                                              |
|+ getC() : double                                                              |
|+ getObjectCount() : int                                                       |
|+ setA(a : double) : void                                                      |
|+ setB(b : double) : void                                                      |
|+ setC(c : double) : void                                                      |
|+ operator+=(n : double) : void                                                |
|+ operator*=(n : double) : void                                                |
|+ operator[](int i) : double                                                   |
|+ operator double()                                                            |
|+ operator<<( Ui::MainWindow *ui, t : Triangle) : void                         |
|+ operator>>(Ui::MainWindow *ui, t) : Triangle : void                          |
|+ operator>(t1 : Triangle, t2 : Triangle) : bool                               |
|+ operator<(t1 : Triangle, t2 : Triangle) : bool                               |
|+ operator==(t1 : Triangle, t2 : Triangle) : bool                              |
`-------------------------------------------------------------------------------'
```

Висновок

На лабораторній роботі я навчився використовувати механізм перевантаження функцій та операцій, створювати та використовувати дружні функції. Ознайомився зі статичними полями і методами та навчився їх використовувати.