МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 5
На тему: "*Створення та використання класів*"
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. Створення та використання класів.

Мета. Навчитися створювати класи, використовувати конструктори для ініціалізації об'єктів, опанувати принципи створення функцій-членів. Навчитися використовувати різні типи доступу до полів та методів класів.

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). Клас повинен містити функції-члени, які реалізовують: а)Знаходження площі трикутника б)Знаходження трьох кутів в)Знаходження периметра г)Знаходження трьох медіан д)Збільшення одразу всіх трьох сторін трикутника на константу е)Задавання значень полів є)Зчитування (отримання значень полів) ж)Перевірка чи трикутник є прямокутний з)Введення трикутника з форми и)Виведення трикутника на форму.

Теоретичні відомості

Ідея класів має на меті дати інструментарій для відображення будови об'єктів реального світу - оскільки кожен предмет або процес має набір характеристик (відмінних рис) іншими словами, володіє певними властивостями і поведінкою. Програми часто призначені для моделювання предметів, процесів і явищ реального світу, тому в мові програмування зручно мати адекватний інструмент для представлення цих моделей. Клас є типом даних, який визначається користувачем. У класі задаються властивості і поведінка будь-якого предмету або процесу у вигляді полів даних (аналогічно до того як це є в структурах) і функцій для роботи з ними. Створюваний тип даних володіє практично тими ж властивостями, що і стандартні типи. Конкретні величини типу даних «клас» називаються екземплярами класу, або об'єктами. Об'єднання даних і функцій їх обробки з одночасним приховуванням непотрібної для використання цих даних інформації називається інкапсуляцією (encapsulation). Інкапсуляція підвищує ступінь абстракції програми: дані класу і реалізація його функцій знаходяться нижче рівня абстракції, і для написання програми з використанням вже готових класів інформації про них (дані і реалізацію функцій) не потрібно. Крім того, інкапсуляція дозволяє змінити реалізацію класу без модифікації основної частини програми, якщо інтерфейс залишився тим самим (наприклад, при необхідності змінити спосіб зберігання даних з масиву на стек). Простота модифікації, як уже неодноразово зазначалося, є дуже важливим критерієм якості програми.

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"

MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow) {
    ui->setupUi(this);
}

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

Triangle t;

void MainWindow::on_setButton_clicked() {
    t = Triangle(ui->aLineEdit->text().toDouble(),
    ui->bLineEdit->text().toDouble(),
    ui->cLineEdit->text().toDouble());
}

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

void MainWindow::on_anglesPushButton_clicked() {
    Angles angles = t.angles();
    ui->anglesResultLabel->setText(QString::fromStdString(" a: ") + QString::
    number(angles.a) +
    QString::fromStdString(" b: ") + QString::number(angles.b) +
    QString::fromStdString(" c: ") + QString::number(angles.c));
}

void MainWindow::on_perimeterPushButton_clicked() {
    ui->perimeterResultLabel->setText(QString::number(t.perimeter()));
}

void MainWindow::on_mediansPushButton_clicked() {
    Medians medians = t.medians();
    ui->mediansResultLabel->setText(QString::fromStdString(" a: ") + QString::
    number(medians.m_a) +
    QString::fromStdString(" b: ") + QString::number(medians.m_b) +
    QString::fromStdString(" c: ") + QString::number(medians.m_c));
}

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

void MainWindow::on_increasePushButton_clicked() {
    t.increaseSides(ui->increaseLineEdit->text().toDouble());
    ui->aLineEdit->setText(QString::number(t.getA()));
    ui->bLineEdit->setText(QString::number(t.getB()));
    ui->cLineEdit->setText(QString::number(t.getC()));
}
```

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_setButton_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();

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

Назва файлу: *triangle.cpp*

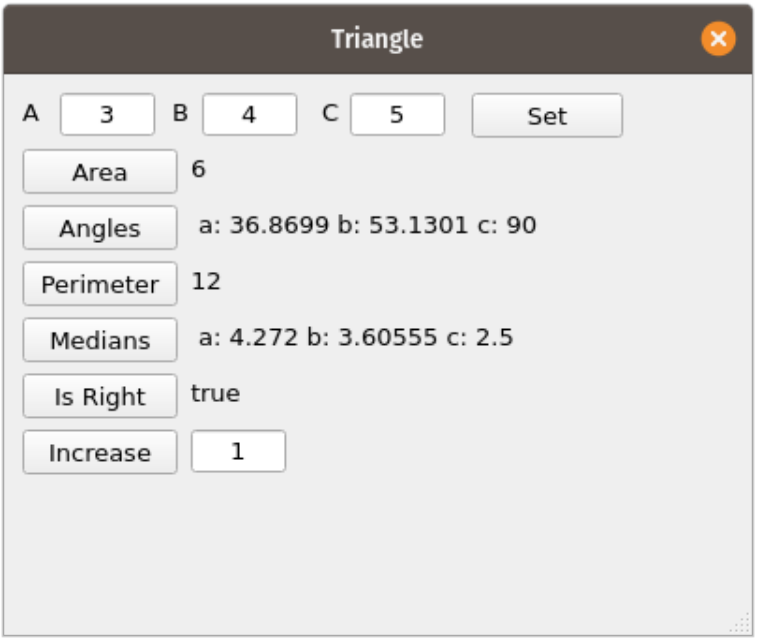
```
#include "triangle.h"
#include <math.h>

double Triangle::area() {
    double p = 0.5 * (m_a + m_b + m_c);
    double S = sqrt(p*(p-m_a)*(p-m_b)*(p-m_c));
    return S;
}

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

double Triangle::perimeter() {
    return m_a + m_b + m_c;
}

Medians Triangle::medians() {
    Medians medians;
    medians.m_a = sqrt((2*m_b*m_b + 2*m_c*m_c - m_a*m_a) / 4);
    medians.m_b = sqrt((2*m_a*m_a + 2*m_c*m_c - m_b*m_b) / 4);
    medians.m_c = sqrt((2*m_a*m_a + 2*m_b*m_b - m_c*m_c) / 4);
    return medians;
}
```

```cpp
int Triangle::isRight() {
    return angles().a == 90.f || angles().b == 90.f || angles().c == 90.f;
}

void Triangle::increaseSides(double n) {
    m_a += n;
    m_b += n;
    m_c += n;
}
```

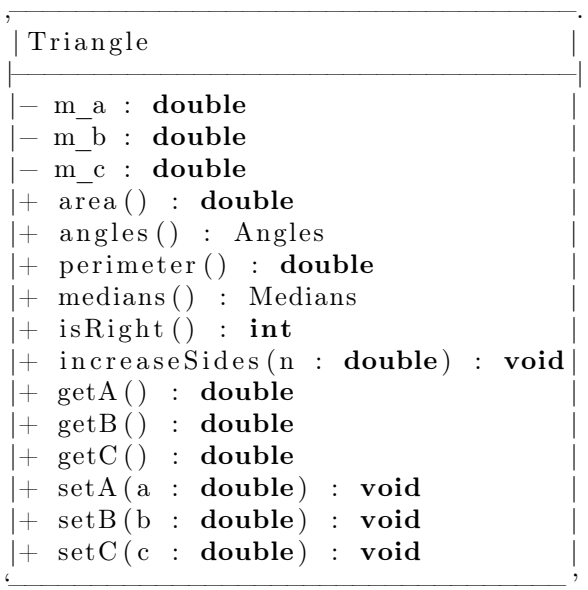
Назва файлу: *triangle.h*

```cpp
#ifndef POLYNOMIAL_H
#define POLYNOMIAL_H

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

struct Medians {
    double m_a;
    double m_b;
    double m_c;
};

class Triangle {
    private:
    double m_a;
    double m_b;
    double m_c;
    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):
    m_a(a),
    m_b(b),
    m_c(c)
    {};

    double area();
    Angles angles();
    double perimeter();
    Medians medians();
    int isRight();
    void increaseSides(double n);

    double getA() const { return m_a; };
    double getB() const { return m_b; };
    double getC() const { return m_c; };
    void setA(double a) { m_a = a; };
    void setB(double b) { m_b = b; };
    void setC(double c) { m_c = c; };
};

#endif // POLYNOMIAL_H
```

Робота програми



UML діаграма



Висновок

На лабораторній роботі я навчився створювати класи, використовувати конструктори для ініціалізації об'єктів, опанувати принципи створення функцій-членів, використовувати різні типи доступу до полів та методів класів.