**Цель работы:** изучение основ объектно-ориентированного программирования, понятий класс, объект, абстракция, инкапсуляция, знакомство с модификаторами доступа, перегрузкой методов, дружественными функциями.

**Постановка задачи.**

Описать класс треугольник, содержащий четыре вещественных поля, три стороны и угол между двумя сторонами, статическое поле для подсчёта количества созданных объектов. Описать два конструктора: конструктор по умолчанию и конструктор с параметрами. Конструктор с параметрами должен содержать три аргумента: две стороны и угол между ними. Написать Get и Set-методы для полей, методы вычисления третей стороны, периметра и площади треугольника. Предусмотреть своевременное вычисление третей стороны, при любом изменении одного из значений полей.

Создать несколько статических и динамических объектов, описанного класса, статический массив объектов, набор объектов используя массив указателей.

Вычислить сумму площадей всех созданных объектов.

Вывести на экран информацию о количестве созданных объектов, содержащуюся в статическом поле.

**Теоретический материал.**

Программный код, написанный по принципам объектно-ориентированного программирования, представляет собой набор абстрактных сущностей, обладающих состоянием и поведением, а также и способов их взаимодействия.

В программировании под классом понимают пользовательский тип, объединяющий в себе разнотипные данные, называемые полями или атрибутами, и функции, обрабатывающие эти данные. Такая функция носит название метод класса и определяет поведение, т. е. способ взаимодействия между экземпляром и другими сущностями, реакции на внешние воздействия экземпляров класса (переменных соответствующего типа).

Для описания класса необходимо указать ключевое слово class, далее имя класса, после чего в фигурных скобках описать все поля данные и методы этого класса. сlass {описание полей и методов класса;};

В качестве примера опишем класс Kvadrat, имеющий одно вещественное поле, хранящее сторону квадрата и метод вычисляющий и возвращающий площадь квадрата, Sq().

class Kvadrat { float a;// поле – сторона квадрата;

public:

Kvadrat(float A) { a = A; };

Kvadrat() { a = 10; };

float Get\_a () { return a; };

void Set\_a(float A){ a = A; };

float Sq() { return a\*a; }; };

Переменные класса, называют экземплярами класса или объектами соответствующего класса.

Объединение данных, описывающих одну сущность и методов для обработки этих данных называют инкапсуляцией. Инкапсуляция – основное свойство ООП.

Данные объекта должны быть скрыты от внешнего использования. Для этой цели служат модификаторы доступа.

Модификаторы доступа – это ключевые слова, после указания которых данные и методы становятся доступны или закрыты для внешнего, т.е. за пределами класса, использования.

Модификатор доступа private запрещает доступ к данным класса за его пределами, public – открывает. Принято поля-данные задавать с модификатором private, иначе происходит нарушение принципа инкапсуляции. Методы же наоборот задают с модификатором public, иначе будет отсутствовать возможность вызова метода за пределами класса.

При необходимости доступа к полям, получения или изменения их значений используют Get, Set методы. О которых будет сказано позднее.

Модификатор доступа protected используют, в случае, когда необходимо защитить данные от внешнего использования, но планируется передать эти данные классам наследникам.

Механизм наследования будет рассмотрен далее.

По умолчанию используется модификатор доступа private. Именно таким образом, в рассмотренном выше примере, поле класса объявлено с модификатором доступа private несмотря на то, что явно этот модификатор не указан.

Если не указать модификатор доступа public явно, то никакие методы нельзя будет вызвать вне класса для его объектов. В том числе и создать объект класса с заданными параметрами. Т. к. при его создании вызывается специфический метод класса, называемый конструктором

Конструктор — это особый метод класса, вызываемый каждый раз при создании объектов этого класса и инициализирующий значения его полей.

Конструктор имеет то же самое имя, что и класс. В заголовочной строке конструктора перед его именем тип возвращаемого значения не указывается, даже void.

**Практическая часть.**

**Triangle.h**

#pragma once

#include <iostream>

using namespace std;

class Triangle

{

private:

double str1;

double str2;

double str3;

double angle;

static int count;

void Find();

public:

Triangle();

Triangle(double str1, double str2, double angle);

double GetAngle();

double GetStr1();

double GetStr2();

double GetP();

double GetS();

void SetAngle(double angle);

void SetStr1(double str1);

void SetStr2(double str2);

void CountTriangl();

~Triangle();

};

**Thriangle.cpp**

#include "Triangle.h"

Triangle::Triangle() {

this->str1 = 3;

this->str2 = 4;

Find();

CountTriangl();

}

Triangle::Triangle(double str1, double str2, double angle) {

this->str1 = str1;

this->str2 = str2;

this->angle = angle;

Find();

CountTriangl();

}

void Triangle::Find() {

str3 = sqrt(str1 \* str1 + str2 \* str2 - 2 \* str1 \* str2 \* cos(angle));

}

double Triangle::GetStr1() {

return str1;

}

double Triangle::GetStr2() {

return str2;

}

double Triangle::GetP() {

Find();

return str1 + str2 + str3;

}

double Triangle::GetS() {

Find();

double S = (str1 \* str2 \* sin(angle) / 2);

return S;

}

double Triangle::GetAngle() {

return angle;

}

void Triangle::SetAngle(double angle) {

if (angle > 0 && angle < 180) {

this->angle = angle;

}

else {

cout << "Угол не может быть меньше 0 и больше 180 градусов или у Вас он не указан." << endl;

this->angle = 52;

}

}

void Triangle::SetStr1(double str1) {

this->str1 = str1;

Find();

}

void Triangle::SetStr2(double str2) {

this->str2 = str2;

Find();

}

void Triangle::CountTriangl() {

++count;

cout << "Создано " << count << " объект(а)" << endl;

}

Triangle::~Triangle() {}

**Lab1.cpp**

#include <iostream>

#include "Triangle.h"

using namespace std;

int Triangle::count;

int main()

{

setlocale(LC\_ALL, "rus");

const int size = 2;

Triangle triangle1;

//с конструктором по умолчанию

cout << "Сторона 1 треугольника = " << triangle1.GetStr1() << endl;

cout << "Сторона 2 треугольника = " << triangle1.GetStr2() << endl;

cout << "Периметр треугольника = " << triangle1.GetP() << endl;

cout << "Площадь треугольника = " << triangle1.GetS() << endl;

triangle1.SetAngle(triangle1.GetAngle());

cout << "Угол между сторонами = " << triangle1.GetAngle() << endl;

cout << "----------------------------" << endl;

//с конструктором с параметрами

Triangle triangle2(4, 5, 90);

cout << "Сторона 1 треугольника = " << triangle2.GetStr1() << endl;

cout << "Сторона 2 треугольника = " << triangle2.GetStr2() << endl;

cout << "Периметр треугольника = " << triangle2.GetP() << endl;

cout << "Площадь треугольника = " << triangle2.GetS() << endl;

triangle1.SetAngle(triangle2.GetAngle());

cout << "Угол между сторонами = " << triangle2.GetAngle() << endl;

//Изменение параметров

triangle2.SetStr1(44);

triangle2.SetStr2(12);

triangle2.SetAngle(152);

cout << "----------------------------" << endl;

cout << "Сторона 1 треугольника = " << triangle2.GetStr1() << endl;

cout << "Сторона 2 треугольника = " << triangle2.GetStr2() << endl;

cout << "Периметр треугольника = " << triangle2.GetP() << endl;

cout << "Площадь треугольника = " << triangle2.GetS() << endl;

cout << "Угол между сторонами = " << triangle2.GetAngle() << endl;

cout << "----------------------------" << endl;

Triangle staticArray[size] = { triangle1, triangle2 };

//нахождение всех площадей

double sumS = 0;

for (int i = 0; i < size; i++) {

sumS += staticArray[i].GetS();

}

cout << "Сумма всех площадей = " << sumS << endl;

Triangle\* triangle3 = new Triangle[size]{};

delete[] triangle3;

}

**Результат выполнения кода.**

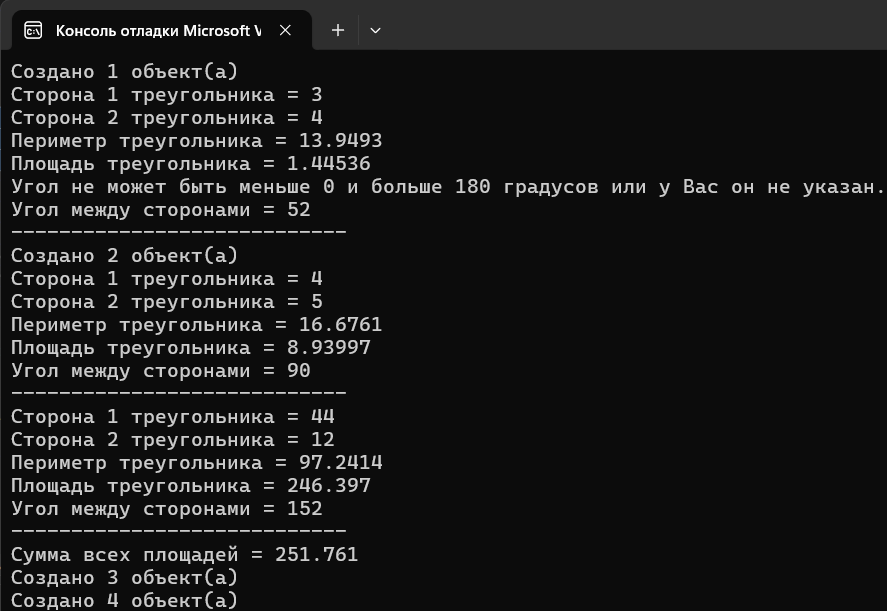


Рисунок 1 – Результат выполнения кода

**Вывод:** в данной лабораторной работе были закреплены навыки работы с классами, навыки работы с конструкторами и деструкторами, а также с определением видимости полей и методов класса.