Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)

Факультет «Робототехника и комплексная автоматизация»

Кафедра «Системы автоматизированного проектирования»

**Отчет по лабораторной работе**

По курсу «Объектно-ориентированное программирование»

Выполнил: Студент Петраков С.А.

Группа РК6-26Б

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2020 г.

**Вариант ПЯВУ К16**

**Задание:**

Разработать объектно-ориентированную программу для вычисления радиуса окружности, в которую вписывается треугольник, заданный координатами своих вершин на плоскости. Эти координаты должны передаваться программе строками потока стандартного ввода. Значения радиуса описанной окружности для каждого заданного треугольника должно отображаться в потоке стандартного вывода. Программная реализация вычислений должна быть основана на разработке контейнерного класса треугольника. Его компоненты-данные должны включать подобъекты класса точки для координат вершин треугольника, а его компонентный метод должен вычислять радиус описанной окружности. Конструкторы классов треугольника и точки должны использовать списки инициализации компонентных данных.

**Алгоритм:**

Для разработки этой программы я буду использовать композитный класс треугольника, в котором будут хранится 3 точки.

1. Вводим 3 точки с координатами x y.
2. Считаем площадь:
   1. Если площадь больше 0, то считаем радиус описанной окружности
   2. Если площадь равна 0, то треугольник вырожден и радиус равен нулю.

**Входные данные:**

Три точки

**Выходные данные:**

Радиус описанной окружности

**Текст программы:**

**main.cpp**

#include <iostream>

#include "Dot.h"

#include "Rectangle.h"

using namespace std;

bool inputFunc(Triangle\* temp)

{

//std::cin.clear();

double x, y;

Dot p[3];

for (int i = 0; i < 3; i++)

{

cout << "Enter point[" << i + 1 << "] coordinats in format: x y." << endl;

cin >> x >> y;

if (std::cin.fail()) //if it return true, previous input failed

return true;

Dot t(x, y);

p[i] = t;

}

Triangle rect(p[0], p[1], p[2]);

\*temp = rect;

return false;

}

int main()

{

Triangle aXzKakNazvat;

cout << "This program counts the radius of the described circle around the triangle." << endl;

while (true)

{

cout << "New triangle:" << endl;

if (inputFunc(&aXzKakNazvat))

{

cout << "Error. Incorrect input. Terminate." << endl << endl;

break;

}

double R;

if (!(R = aXzKakNazvat.calcRadius()))

cout << "Error. The triangle doesn't exist. R=0." << endl;

else

cout << "Described circle around this triangle is " << R << "." << endl << endl;

}

return 0;

}

**Rectangle.h**

#pragma once

#ifndef RECTANGLEH

#define RECTANGLEH

#include "Dot.h"

class Triangle

{

private:

Dot point[3];

public:

Triangle();

Triangle(Dot A, Dot B, Dot C);

double calcRadius();

double calcLine(int, int);

};

#endif

**Rectangle.cpp**

#include "Rectangle.h"

#include "Dot.h"

#include <math.h>

#define EPS 1e-7

Triangle::Triangle() {};

Triangle::Triangle(Dot A, Dot B, Dot C) :point{ A,B,C } {}

double Triangle::calcRadius()

{

double S = abs((point[0].\_x - point[2].\_x) \* (point[1].\_y - point[2].\_y) - (point[1].\_x - point[2].\_x) \* (point[0].\_y - point[2].\_y)) / 2;

if (S < EPS)

return 0;

double R, a, b, c;

a = calcLine(0, 1);

b = calcLine(0, 2);

c = calcLine(1, 2);

R = a \* b \* c / (4 \* S);

return R;

}

double Triangle::calcLine(int a, int b)

{

return sqrt(pow(point[a].\_x-point[b].\_x,2)+ pow(point[a].\_y - point[b].\_y, 2));

}

**Dot.h**

#pragma once

#ifndef DOTH

#define DOTH

#include <iostream>

using namespace std;

class Dot

{

friend class Triangle;

private:

double \_x;

double \_y;

public:

double getx();

double gety();

Dot(double, double);

Dot();

};

#endif

**Dot.cpp**

#include "Dot.h"

double Dot::getx()

{

return \_x;

}

double Dot::gety()

{

return \_y;

}

Dot::Dot(double x, double y) :\_x(x), \_y(y) {}

Dot::Dot()

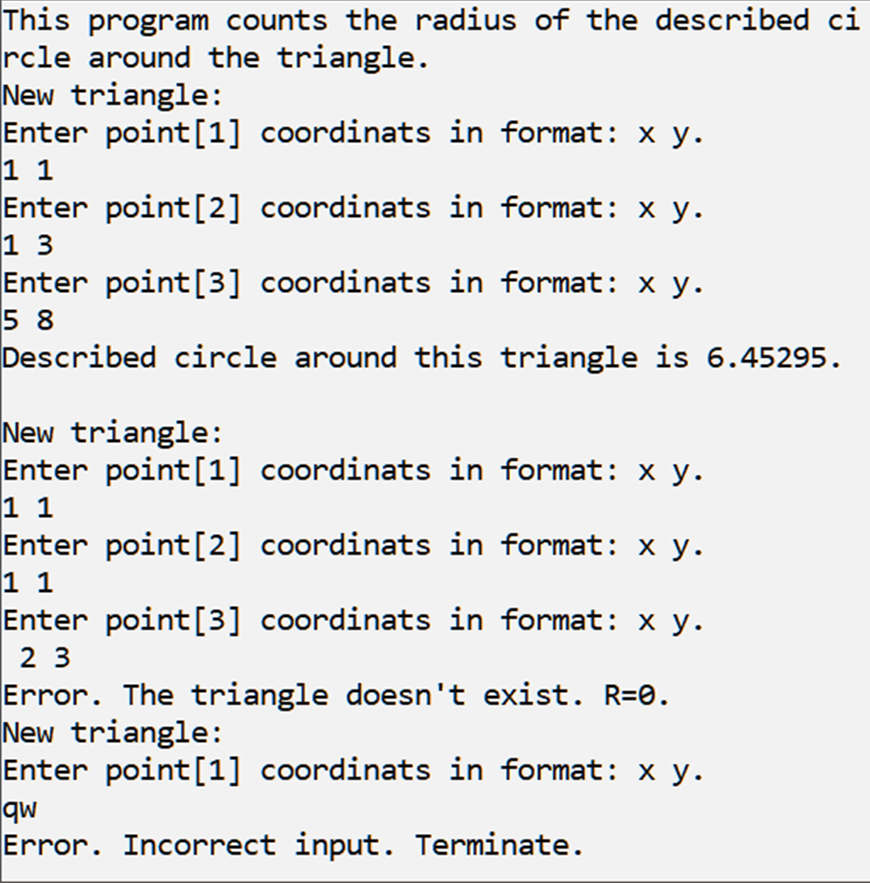
{

\_x = 0;

\_y = 0;

}

**Тесты:**

****

**Список использованной литературы:**

* Волосатова Т.М., Родионов С.В. Лекции по курсу «Объектно-ориентированное программирование»
* bigor.bmstu.ru