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 Сергеева Д.К.

Группа РК6-26Б

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

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2020 г.

**ПЯВУ 20К**

**Задание:**

Разработать объектно-ориентированную программу для определения углов треугольников, которые задаются координатами вершин на плоскости. Координаты вершин треугольников должны передаваться программе строками потока стандартного ввода. Значения углов для каждого заданного треугольника должны отображаться через поток стандартного вывода. Программная реализация вычислений должна быть основана на разработке контейнерного класса треугольника. Его компоненты-данные должны включать подобъекты класса точки для хранения координат вершин треугольника, а компонентные методы должны определять по ним значения углов треугольника. Конструкторы классов треугольника и точки должны использовать списки инициализации компонентных данных.

**Алгоритм:**

Создаем композитный класс треугольника, который хранит классы 3 точек. Сначала проверяем, что треугольник существует, то есть проверяем, что 3 точки не лежат на 1 прямой. Для этого считаем площадь треугольника, если она равна 0, то это треугольник не существует, иначе считаем все углы треугольника, используя координаты его вершин, по формуле через cos ().

**Входные данные:**

Координаты 3 вершин треугольника.

**Выходные данные:**

Значения 3 углов треугольника.

**Текст программы:**

#include <iostream>

#include <cmath>

using namespace std;

class Triangle;

class Dot

{

private:

double \_x;

double \_y;

public:

Dot(double x = 0.0, double y = 0.0) : \_x(x), \_y(y) {};

friend class Triangle;

};

class Triangle

{

private:

Dot A;

Dot B;

Dot C;

public:

Triangle(Dot d1, Dot d2, Dot d3) : A(d1), B(d2), C(d3) {};

Triangle() {};

double angleABC();

double angleBCA();

double angleCAB();

bool exist\_triangle();

};

bool Triangle::exist\_triangle()

{

if ((((A.\_x - C.\_x) \* (B.\_y - C.\_y) - (B.\_x - C.\_x) \* (A.\_y - C.\_y)) / 2) == 0)

return false;

else

return true;

}

double Triangle::angleABC()

{

double x1 = (B.\_x - A.\_x), x2 = (B.\_x - C.\_x), y1 = (B.\_y - A.\_y), y2 = (B.\_y - C.\_y);

double sqrt1 = sqrt(x1 \* x1 + y1 \* y1), sqrt2 = sqrt(x2 \* x2 + y2 \* y2);

return acos((x1\*x2 + y1 \* y2) / sqrt1 / sqrt2) \* 180.0 / 3.14159265;

}

double Triangle::angleBCA()

{

double x1 = (C.\_x - B.\_x), x2 = (C.\_x - A.\_x), y1 = (C.\_y - B.\_y), y2 = (C.\_y - A.\_y);

double sqrt1 = sqrt(x1 \* x1 + y1 \* y1), sqrt2 = sqrt(x2 \* x2 + y2 \* y2);

return acos((x1\*x2 + y1 \* y2) / sqrt1 / sqrt2) \* 180.0 / 3.14159265;

}

double Triangle::angleCAB()

{

double x1 = (A.\_x - B.\_x), x2 = (A.\_x - C.\_x), y1 = (A.\_y - B.\_y), y2 = (A.\_y - C.\_y);

double sqrt1 = sqrt(x1 \* x1 + y1 \* y1), sqrt2 = sqrt(x2 \* x2 + y2 \* y2);

return acos((x1\*x2 + y1 \* y2) / sqrt1 / sqrt2) \* 180.0 / 3.14159265;

}

bool input\_coord(Triangle\* Tr)

{

double x, y;

Dot point[3];

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

{

cout << "Input 2 coordinates of dot " << (char)(65 + i) << " in format: \"x y\" " << endl;

if (cin >> x >> y)

{

Dot temp(x, y);

point[i] = temp;

}

else

return false;

}

Triangle Tr\_temp(point[0], point[1], point[2]);

\*Tr = Tr\_temp;

return true;

}

int main()

{

Triangle ABC;

while (input\_coord(&ABC))

{

if (!(ABC.exist\_triangle()))

{

cout << "Dots lie on 1 line or dots matches. Triangle don't exist." << endl;

return 0;

}

cout << "Angle ABC = " << ABC.angleABC() << " degree" << endl;

cout << "Angle BCA = " << ABC.angleBCA() << " degree" << endl;

cout << "Angle CAB = " << ABC.angleCAB() << " degree" << endl;

return 0;

}

cout << "Wrong input of coordinates" << endl;

return 0;

}

**Тесты:**

|  |  |  |
| --- | --- | --- |
| **Номер** | **Вход** | **Выход** |
| 1 | 1 1  1 1  3 0 | Dots lie on 1 line or dots matches. Triangle don't exist. |
| 2 | -1 0  0 1  1 0 | Angle ABC = 90 degree  Angle BCA = 45 degree  Angle CAB = 45 degree |

**Список использованной литературы:**

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