ФИО: Гольчанский Максим Дмитриевич

Группа: 3530904/10003

Вариант №2

Дата сдачи: 26.11.2021

#include <iostream>

#include <exception>

#include <cmath>

#ifndef POINT\_H

#define POINT\_H

class Point

{

private:

float x\_, y\_;

public:

Point();

Point(float, float);

Point(const Point& p) : x\_(p.x\_), y\_(p.y\_) {}

~Point() {};

void setX(float);

void setY(float);

float getX() const;

float getY() const;

bool isEqual(const Point&) const;

float getDistance(const Point&) const;

void move(float);

};

#endif

// Point.cpp

#include "Point.h"

Point::Point() {

x\_ = 0.f;

y\_ = 0.f;

}

Point::Point(float x, float y) {

x\_ = x;

y\_ = y;

}

void Point::setX(float x) { x\_ = x; }

void Point::setY(float y) { y\_ = y; }

float Point::getX() const { return x\_; }

float Point::getY() const { return y\_; }

bool Point::isEqual(const Point& other) const { return other.getX() == x\_ && other.getY() == y\_; }

float Point::getDistance(const Point& other) const {

return std::sqrt(

(x\_ - other.getX()) \* (x\_ - other.getX()) +

(y\_ - other.getY()) \* (y\_ - other.getY())

);

}

void Point::move(float K) {

x\_ += K;

y\_ += K;

}

// Triangle.h

#include "Point.h"

#ifndef TRIANGLE\_H

#define TRIANGLE\_H

class Triangle

{

private:

Point a\_, b\_, c\_;

public:

Triangle();

Triangle(const Point&, const Point&, const Point&);

Triangle(const Triangle& t) : a\_(t.a\_), b\_(t.b\_), c\_(t.c\_) {}

~Triangle() {}

void setA(const Point&);

void setB(const Point&);

void setC(const Point&);

const Point& getA() const;

const Point& getB() const;

const Point& getC() const;

bool isTriangle() const;

void move(float);

float getPerimeter() const;

bool isEqual(const Triangle&) const;

};

#endif

// Triangle.cpp

#include "Triangle.h"

Triangle::Triangle() {

a\_ = Point();

b\_ = Point();

c\_ = Point();

}

Triangle::Triangle(const Point& a, const Point& b, const Point& c) {

a\_ = a;

b\_ = b;

c\_ = c;

}

void Triangle::setA(const Point& a) { a\_ = a; }

void Triangle::setB(const Point& b) { b\_ = b; }

void Triangle::setC(const Point& c) { c\_ = c; }

const Point& Triangle::getA() const { return a\_; }

const Point& Triangle::getB() const { return b\_; }

const Point& Triangle::getC() const { return c\_; }

bool Triangle::isTriangle() const {

float a = a\_.getDistance(b\_);

float b = b\_.getDistance(c\_);

float c = c\_.getDistance(a\_);

return a + b < c&& b + c < a&& a + c < b;

}

void Triangle::move(float K) {

a\_.move(K);

b\_.move(K);

c\_.move(K);

}

float Triangle::getPerimeter() const {

float a = a\_.getDistance(b\_);

float b = b\_.getDistance(c\_);

float c = c\_.getDistance(a\_);

return a + b + c;

}

bool Triangle::isEqual(const Triangle& t) const {

return a\_.isEqual(t.getA()) && b\_.isEqual(t.getB()) && c\_.isEqual(t.getC());

}

// main.cpp

#include <iostream>

#include <exception>

#include <cmath>

bool isPointInCircle(const Point& point, float radius) {

return point.getX() \* point.getX() + point.getY() \* point.getY() <= radius \* radius;

}

void input(Point& point) {

float x = 0.f;

float y = 0.f;

std::cout << "Введите x, y: ";

std::cin >> x >> y;

point.setX(x);

point.setY(y);

}

void output(const Point& point) {

std::cout << "Point {" << point.getX() << ", " << point.getY() << "};\n";

}

void Output(const Triangle& triangle) {

std::cout << "Triangle {\n"

<< "\t" << triangle.getA().getX() << ", " << triangle.getA().getY() << "\n"

<< "\t" << triangle.getB().getX() << ", " << triangle.getB().getY() << "\n"

<< "\t" << triangle.getC().getX() << ", " << triangle.getC().getY() << "\n"

<< "};\n";

}

bool isEqualSquare(const Triangle& first, const Triangle& second) {

float a1 = first.getA().getDistance(first.getB());

float b1 = first.getB().getDistance(first.getC());

float c1 = first.getC().getDistance(first.getA());

float p1 = first.getPerimeter() / 2;

float s1 = std::sqrt(p1 \* (a1 - p1) \* (b1 - p1) \* (c1 - p1));

float a2 = second.getA().getDistance(second.getB());

float b2 = second.getB().getDistance(second.getC());

float c2 = second.getC().getDistance(second.getA());

float p2 = second.getPerimeter() / 2;

float s2 = std::sqrt(p2 \* (a2 - p2) \* (b2 - p2) \* (c2 - p2));

if (s1 == s2) {

return true;

}

else {

return false;

}

}

void BuildTriangle(Triangle& triangle) {

Point b;

input(b);

Point a = b;

int count = 1;

while (count < 4) {

a = b;

input(b);

triangle.setA(a);

count++;

if (count == 2) {

triangle.setB(a);

}

if (count == 3) {

triangle.setC(a);

}

}

Output(triangle);

}

int main() {

Triangle t;

Triangle t2;

setlocale(LC\_ALL, "ru");

BuildTriangle(t);

BuildTriangle(t2);

if (isEqualSquare(t, t2) == true) {

std::cout << "Площади треугольников равны";

}

else {

std::cout << "Площади треугольников не равны";

}

}