|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация (РК)

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**по курсу объектно-ориентированное программирование**

Студент Камалов Антон Павлович

Группа РК6-25Б

Тип задания Лабораторная работа №5

Вариант 19К

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Камалов А.П.**

*подпись, дата фамилия, и.о.*

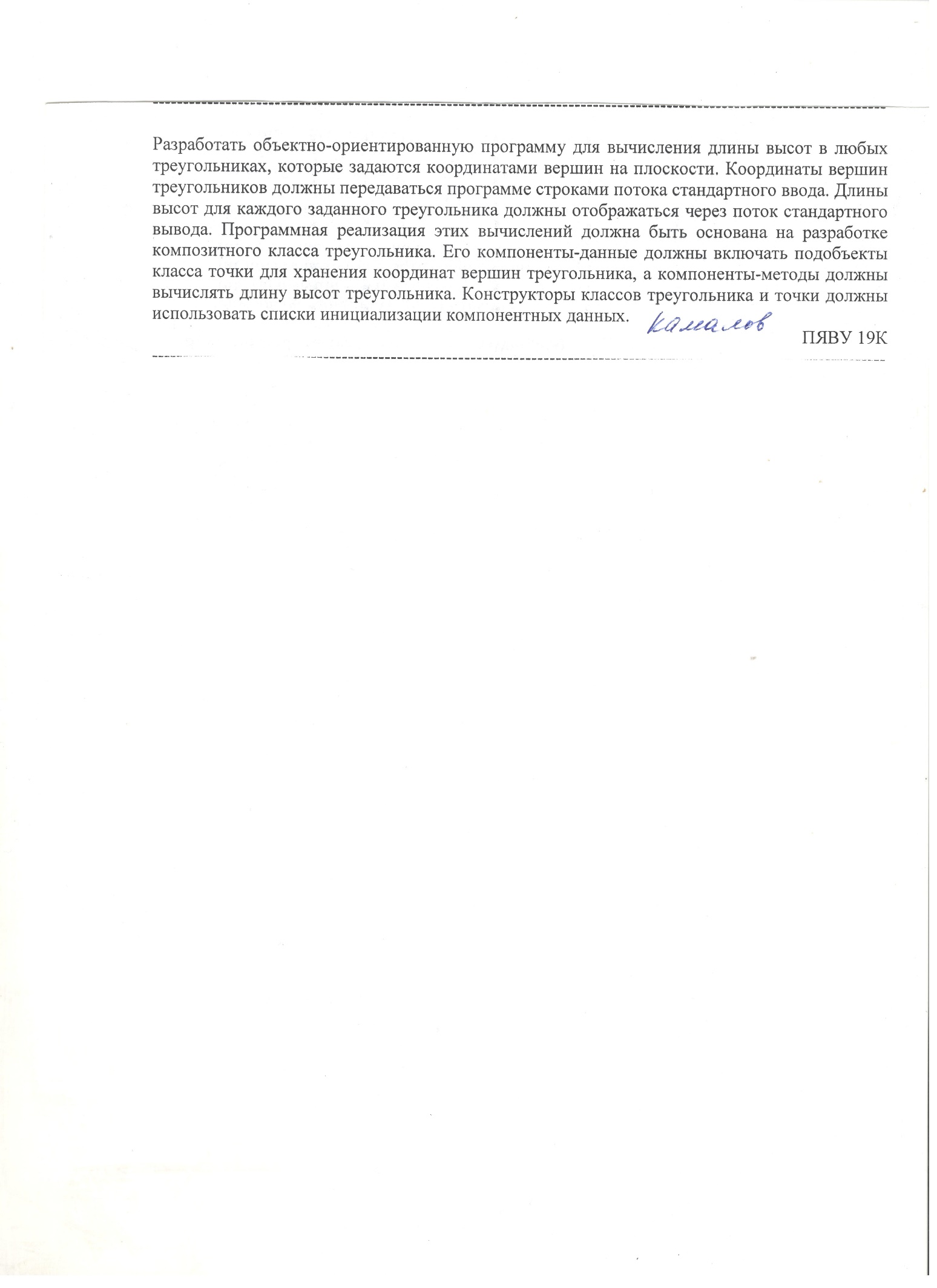
Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кузьмина И.А.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2023 г.*

**Задание**

****

**Описание программы**

**Входные данные:** координат вершин треугольника.

**Выходные данные:** длины высот треугольника.

**Класс Point**

*Информационные поля:*

\_x (double) – координаты точки по оси X;

\_y (double) – координаты точки по оси Y;

*Конструкторы и деструкторы:*

Point (double, double) – конструктор со списком инициализации;

Point () – конструктор по умолчанию;

*Методы:*

getX () – публичный метод получения координаты точки по оси X;

getY() – публичный метод получения координаты точки по оси Y;

friend std::istream& operator char << (std::istream&, Point&) – публичный метод перегрузки оператора побитового сдвига вправо;

**Класс Triangle**

*Информационные поля:*

p1 (Point) – координаты первой вершины треугольника;

p2 (Point) – координаты второй вершины треугольника;

p3 (Point) – координаты третьей вершины треугольника;

*Конструкторы и деструкторы:*

Triangle (Point&, Point&, Point&) – конструктор со списком инициализации;

*Методы:*

tr\_area (double, double, double) – приватный метод вычисления площади треугольника;

distance (Point&, Point&) – приватный метод расстояния между вершина треугольника;

HeightLengths() – публичный метода вывод длин высот треугольника на экран.

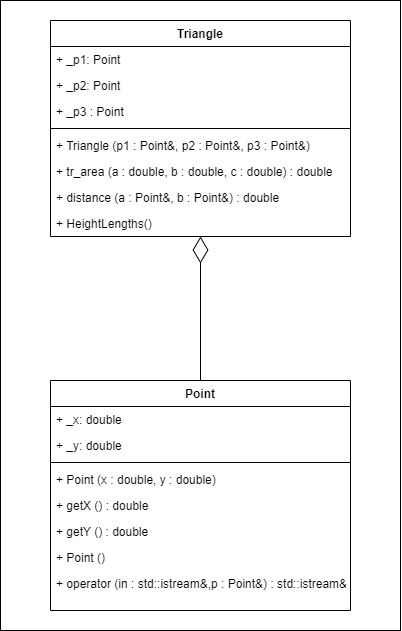


Рисунок 1. Диаграмма классов

**Алгоритм решения:**

1. Создаём 3 объека класса Point – p1, p2, p3 – которые будут хранить координаты вершин треугольника;
2. Вводим координаты вершин треугольника;
3. Создаём объект класса Triangle – tr.
4. Вызываем функцию HeightLengths().

Для вычисления длин высот, нам потребуется вычислить стороны треугольника, а также его площадь.

Площадь вычисляется по формуле Герона с помощью функции tr\_area():

,

где – полупериметр.

Длина стороны треугольника определяется с помощью функции distance() по следующей формуле:

,

где x1, y1, x2, y2 – координаты двух вершин треугольника.

Если одна из сторон оказывается равна нулю, то программа завершается с кодом ошибки -1.

В другом случае выводим на экран длины высот треугольника по формуле:

,

где a – длина стороны, на которую падает высота.

1. Повторяем шаги 2 – 4 до тех пор, пока пользователь не выйдет из программы или длина одной из сторон не окажется равной 0.

**Список литературы**

1. Пол Айра. Объектно-ориентированное программирование на C++: пер. с англ. 2-е изд. СПБ.: Невский Диалект; М.: Издательство БИНОМ, 2001. 462 с. [Pohl Ira. Object-Oriented Programming Using C++. 2nd ed. Addison-Wesley, 1996. 576 p.].
2. Страуструп Б. Язык программирования C++: пер. с англ / под ред. Н.Н. Мартынова. Специальное изд. М.: Бином, 2011. 1035 c. [Stroustrup B. The C++ Programming Language. Special ed. Addison-Wesley, 2000. 1029p.].
3. Шилдт Г. Самоучитель C++. 3-е изд. СПБ.: БХВ-Петербург, 2002. 688 с. [Schildt H. Teach Yourself C++. 3d ed. McGraw-Hill, 1998. 768 p.].
4. Волосатова Т.М., Родионов С.В. Объектно-ориентированное программирование на С++. Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=VU/base.cou (дата обращения 17.04.2023).

**Приложение 1**

Текст программы

#include <iostream>

#include <cstdio>

#include <cmath>

class Point {

private:

double \_x;

double \_y;

public:

Point() : \_x(0), \_y(0) {};

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

double getX() {return \_x ;}

double getY() {return \_y ;}

friend std::istream& operator >>(std::istream&, Point&);

};

class Triangle {

private:

Point \_p1;

Point \_p2;

Point \_p3;

double tr\_area(double, double, double);

double distance(Point&, Point&);

public:

Triangle(Point& p1, Point& p2, Point& p3): \_p1(p1), \_p2(p2), \_p3(p3) {}

void HeightLengths();

};

std::istream& operator >>(std::istream& in, Point& p){

char sep = ';';

in >> p.\_x >> sep >> p.\_y;

return in;

}

double Triangle::distance(Point& p1, Point& p2){

double dx = p1.getX() - p2.getX();

double dy = p1.getY() - p2.getY();

return sqrt(dx \* dx + dy \* dy);

}

double Triangle::tr\_area(double a, double b, double c){

double semi = (a + b + c)/2;

return sqrt(semi \* (semi - a) \* (semi - b) \* (semi - c));

}

void Triangle::HeightLengths(){

double a = distance(\_p1, \_p2);

double b = distance(\_p1, \_p3);

double c = distance(\_p2, \_p3);

double area = tr\_area(a, b, c);

if (a == 0 || b == 0 || c == 0){

std::cout << "Error. Exit!\n";

exit(-1);

}

printf("First height - %.3lf \nSecond height - %.3lf \nThird height - %.3lf\n\n", 2\*area/a, 2\*area/b, 2\*area/c);

}

int main(){

Point p1, p2, p3;

std::cout << "Input x1;y1 x2;y2 x3;y3 or exit:\n";

while(std::cin >> p1 >> p2 >> p3){

Triangle tr(p1, p2, p3);

tr.HeightLengths();

std::cout << "Input x1;y1 x2;y2 x3;y3 or exit:\n";

}

}

**Приложение 2**

Результат работы программы

