**САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО**

**ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И КИБЕРБЕЗОПАСНОСТИ**

**ВЫСШАЯ ШКОЛА ПРОГРАММНОЙ ИНЖЕНЕРИИ**

**Лабораторная работа №8**

**«ООП. Перегрузка операторов»**

**по предмету «Алгоритмизация и программирование»**

**Выполнил: студент гр. 5130904/30002 Севостьянова А.В.**

**Руководитель: Череповский Д.К.**

**Санкт-Петербург**

**2023 г.**

Оглавление

[1. Общая постановка задачи 3](#_Toc152369712)

[Задание 1 3](#_Toc152369713)

[Задание 2 3](#_Toc152369714)

[2. Детальные требования и тест план 4](#_Toc152369715)

[1. Все переменные, вводимые с консоли (значения Х и Y точек, расстояние, на которое будет перемещена точка или треугольник, радиус окружности), должны быть заданы дробными числам: 4](#_Toc152369716)

[2. Для нахождения периметра заданные точки должны образовывать треугольник: 4](#_Toc152369717)

[3. Для сравнения площадей обе заданные фигуры должны являться треугольниками: 4](#_Toc152369718)

[Таблица с детальными требованиями и тест планом 4](#_Toc152369719)

[3. Программа 6](#_Toc152369720)

[4. Порядок вызова конструкторов и деструкторов 12](#_Toc152369721)

[Приложение 13](#_Toc152369722)

[Вывод 14](#_Toc152369723)

1. **Общая постановка задачи**

## *Задание 1*

Для класса Point перегрузите следующие операторы.

*Методы класса:*

* operator=
* operator== (a == b, true, если равны координаты)
* operator< (a < b, true, если сумма квадратов координат точки a меньше суммы квадратов координат b)
* operator>= (a >= b, true, если сумма квадратов координат точки a не меньше суммы квадратов координат b)
* operator+ (a + k, сложение координат точки a с заданным числом k)
* operator+= (a += k, сложение координат точки a с заданным числом k и сохранение значения в a.)

*Дружественные функции:*

1) operator<<

2) operator>>

***Задание 2***

Для класса Triangle (см. ООП. Тема 1. Задание 2) перегрузите следующие операторы.

*Методы класса:*

* operator=
* operator==
* t1 == t2, (true, если равны координаты всех вершин треугольников. Порядок обхода вершин (a\_, b\_, c\_) )
* operator< (t1 < t2, true, если площадь треугольника t1, меньше площади треугольника t2)
* operator+ (t + k, сложение координат вершин t с заданным числом k)
* operator+= (t += k, сложение координат вершин t с заданным числом k и сохранение значения в t)

*Дружественные функции:*

1) operator<<

2) operator>>

1. **Детальные требования и тест план**

## *1. Все переменные, вводимые с консоли (значения Х и Y точек, расстояние, на которое будет перемещена точка или треугольник, радиус окружности), должны быть заданы дробными числам:*

* Если переменная принимает нечисловое значение, то сообщение: «Некорректный ввод данных», аварийное завершение программы

## *2. Для нахождения периметра заданные точки должны образовывать треугольник:*

* Если заданные точки не образуют треугольник (метод isTriangle вернул ложное значение), то сообщение «Фигура не является треугольником», аварийное завершение работы программы.

## *3. Для сравнения площадей обе заданные фигуры должны являться треугольниками:*

* Если заданные точки не образуют треугольник хотя бы для одной из переданных в качестве параметров функции фигур (метод isTriangle вернул ложное значение), то сообщение «Фигура не является треугольником», аварийное завершение работы программы.

## *Таблица с детальными требованиями и тест планом*

|  |  |  |  |
| --- | --- | --- | --- |
| Требования | Детальные требования | Данные | Ожидаемый результат |
| 1. ***Вводимые с консоли переменные должны быть заданы корректно*** | | | |
| 1. Переменная - число | Если переменная принимает нечисловое значение, то сообщение:  «Некорректный ввод данных», аварийное завершение программы | rtryu | Сообщение:  «Некорректный ввод данных»  Аварийное завершение программы. |
| 2. Переменная задана корректно | Если переменная задана корректно, то продолжение работы программы | 0.5  0  7 | Продолжение работы программы |
| 1. ***При нахождение периметра фигура должна быть треугольником*** | | | |
| 1. Заданные точки не образуют треугольник | Если заданные точки не образуют треугольник (метод isTriangle вернул ложное значение), то сообщение «Фигура не является треугольником», аварийное завершение работы программы. |  | Сообщение: «Фигура не является треугольником»  Аварийное завершение работы программы. |
| 2. Заданные точки образуют треугольник | Если заданные точки образуют треугольник (метод isTriangle вернул положительное значение), то определение периметра треугольника. Продолжение работы программы. |  | Продолжение работы программы. |
| 1. ***При сравнении площадей обе фигуры должны являться треугольниками*** | | | |
| 1. Хотя бы 1 из заданных фигур не является треугольником | Если заданные точки не образуют треугольник хотя бы для одной из переданных в качестве параметров функции фигур (метод isTriangle вернул ложное значение), то сообщение «Фигура не является треугольником», аварийное завершение работы программы. |  | Сообщение: «Фигура не является треугольником»  Аварийное завершение работы программы. |
| 2. Если обе фигуры являются треугольниками | Если точки обеих фигур образуют треугольники, то сравнение их площадей |  | Продолжение работы программы |

1. **Программа**

***main.cpp***

#include"Point.h";

#include"Triangle.h"

int main()

{

setlocale(LC\_ALL, "ru");

try {

testPoint();

tetsTriangle();

}

catch (std::invalid\_argument& ex)

{

std::cerr << ex.what();

exit(EXIT\_FAILURE);

}

catch (std::runtime\_error& ex)

{

std::cerr << ex.what();

exit(EXIT\_FAILURE);

}

return 0;

}

***Point.h***

#pragma once

#include<iostream>

#include<cmath>

#include<regex>

const std::string ERROR\_OF\_INPUT = "Некорректный ввод данных\n";

class Point

{

//перегрузка операторов ввода и вывода

friend std::ostream& operator << (std::ostream& stream, const Point& point);

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

public:

Point();

Point(double x, double y);

Point(const Point& a);

~Point();

void setX(double x);

void setY(double y);

double getX()const;

double getY() const;

double getDistance(const Point a)const;

//перегрузка операторов сравнения и присваивания

bool operator == (const Point& point) const;

bool operator < (const Point& point) const;

bool operator >= (const Point& point) const;

Point& operator = (const Point& point);

Point operator + (const double k)const;

Point& operator += (const double k);

private:

double x\_,

y\_;

};

double checkInput(std::string& value);

void iSpointInCircle(const Point a, const double r);

void testPoint();

***Point.cpp***

#include "Point.h"

Point::Point() { x\_ = 0.0; y\_ = 0.0; }

Point::Point(double x, double y) { x\_ = x; y\_ = y; }

Point::Point(const Point& a) { x\_ = a.x\_; y\_ = a.y\_; }

Point::~Point() { }

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

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

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

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

double Point::getDistance(const Point a) const

{

return sqrt(pow(x\_ - a.x\_, 2) + pow(y\_ - a.y\_, 2));

}

void iSpointInCircle(const Point a, const double r)

{

sqrt(pow(a.getX(), 2) + pow(a.getY(), 2)) <= r ? std::cout << "Точка попадает в круг радиуса " << r << "\n" :

std::cout << "Точка не попадает в круг радиуса " << r << "\n";

}

// перегрузка операторов сравнения

bool Point::operator == (const Point& point) const { return (x\_==point.x\_) && (y\_==point.y\_);}

bool Point::operator < (const Point& point) const { return (pow(x\_,2)+pow(y\_,2)) < (pow(point.x\_, 2) + pow(point.y\_, 2)); }

bool Point::operator >= (const Point& point) const { return (pow(x\_, 2) + pow(y\_, 2)) >= (pow(point.x\_, 2) + pow(point.y\_, 2)); }

//перегрузка операторов присваивания

Point& Point::operator = (const Point& point)

{

x\_ = point.x\_;

y\_ = point.y\_;

return \*this;

}

Point Point::operator + (const double k)const

{

Point point(x\_ + k, y\_ + k);

return point;

}

Point& Point::operator += (const double k)

{

x\_ += k;

y\_ += k;

return \*this;

}

//перегрузка операторов ввода и вывода

double checkInput(std::string& value)

{

std::regex regular("-?\\d+,?\\d\*");

std::regex replace("\\.");

value = std::regex\_replace(value, replace, ",");

if (std::regex\_match(value, regular)) return std::stod(value);

else throw std::invalid\_argument(ERROR\_OF\_INPUT);

}

std::istream& operator >> (std::istream& stream, Point& point)

{

std::string value1 = "";

std::string value2 = "";

std::cout << "координаты точки:\t";

stream >> value1 >> value2;

point.x\_ = checkInput(value1);

point.y\_ = checkInput(value2);

return stream;

}

std::ostream& operator << (std::ostream& stream, const Point& point)

{

stream << "(" << point.x\_ << ", " << point.y\_ << ")";

return stream;

}

// функция, тестирующая класс Point

void testPoint()

{

std::cout << "Тестирование класса Point\n";

Point point1;

Point point2;

std::cin >> point1;

std::cin >> point2;

point1 == point2 ? std::cout << "Точки совпадают\n" : std::cout << "Точки не совпадают\n";

if (point1 < point2) std::cout << point1 << " < " << point2 << "\n";

if (point1 >= point2) std::cout << point1 << " >= " << point2 << "\n";

std::cout << "Введите расстояние, на которое будут перемещены точки: ";

std::string value = "";

std::cin >> value;

point1 = point1 + checkInput(value);

point2 += checkInput(value);

std::cout << "Координаты точек:" << point1 << "\t" << point2 << "\n";

std::cout << "Конец тестирования класса Point\n\n\n";

}

***Triangle.h***

#pragma once

#include"Point.h"

const std::string ERROR\_OF\_FIGURE = "Фигура не является треугольником\n";

class Triangle

{

//перегрузка операторов ввода и вывода

friend std::ostream& operator << (std::ostream& stream, const Triangle& triangle);

friend std::istream& operator >> (std::istream& stream, Triangle& triangle);

public:

Triangle();

Triangle(Point a, Point b, Point c);

Triangle(const Triangle& a);

~Triangle();

void setA(Point a);

void setB(Point b);

void setC(Point c);

Point getA() const;

Point getB() const;

Point getC() const;

bool isTriangle()const;

double getPerimeter() const;

//перегрузка операторов сравнения и присваивания

bool operator == (const Triangle& triangle) const;

bool operator < (const Triangle& triangle) const;

Triangle operator + (const double k) const;

Triangle& operator += (const double k);

private:

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

};

void tetsTriangle();

***Triangle.cpp***

#include "Triangle.h"

Triangle::Triangle()

{

a\_.setX(0.0); a\_.setY(0.0);

b\_.setX(0.0); b\_.setY(0.0);

c\_.setX(0.0); c\_.setY(0.0);

}

Triangle::Triangle(Point a, Point b, Point c)

{

a\_ = a;

b\_ = b;

c\_ = c;

}

Triangle::Triangle(const Triangle& a)

{

a\_ = a.getA();

b\_ = a.getB();

c\_ = a.getC();

}

Triangle::~Triangle() {};

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

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

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

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

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

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

bool Triangle::isTriangle() const

{

double AB = a\_.getDistance(b\_);

double AC = a\_.getDistance(c\_);

double BC = b\_.getDistance(c\_);

return ((AB + BC > AC) && (AB + AC > BC) && (BC + AC > AB)) ? EXIT\_FAILURE : EXIT\_SUCCESS;

}

double Triangle::getPerimeter() const

{

double AB = a\_.getDistance(b\_);

double AC = a\_.getDistance(c\_);

double BC = b\_.getDistance(c\_);

if (isTriangle()) return AB+BC+AC;

else throw std::runtime\_error(ERROR\_OF\_FIGURE);

}

//пергрузка операторов сравнения

bool Triangle::operator==(const Triangle& triangle) const

{

if (isTriangle() && triangle.isTriangle())

return (a\_ == triangle.a\_) && (b\_ == triangle.b\_) && (c\_ == triangle.c\_);

else throw std::runtime\_error(ERROR\_OF\_FIGURE);

}

bool Triangle::operator<(const Triangle& triangle) const

{

double AB = a\_.getDistance(b\_);

double AC = a\_.getDistance(c\_);

double BC = b\_.getDistance(c\_);

double square1 = sqrt(0.5\*getPerimeter()\*(0.5 \* getPerimeter()-AB)\*(0.5 \* getPerimeter()-BC)\*(0.5 \* getPerimeter()-AC));

AB = triangle.a\_.getDistance(triangle.b\_);

AC = triangle.a\_.getDistance(triangle.c\_);

BC = triangle.b\_.getDistance(triangle.c\_);

double square2 = sqrt(0.5 \* triangle.getPerimeter() \* (0.5 \* triangle.getPerimeter() - AB) \* (0.5 \* triangle.getPerimeter() - BC) \* (0.5 \* triangle.getPerimeter() - AC));

return square1 < square2;

}

//перегрузка операторов присваивания

Triangle Triangle::operator + (const double k) const

{

Triangle new\_triangle(a\_+k,b\_+k,c\_+k);

return new\_triangle;

}

Triangle& Triangle::operator+=(const double k)

{

a\_ += k;

b\_ += k;

c\_ += k;

return \*this;

}

//перегрузка операторов ввода и вывода

std::ostream& operator<<(std::ostream& stream, const Triangle& triangle)

{

stream << "Координаты треугольника: " << triangle.a\_ << " " << triangle.b\_ <<" "<<triangle.c\_<< "\n";

return stream;

}

std::istream& operator>>(std::istream& stream, Triangle& triangle)

{

std::cout << "1 вершина:\t";

stream >> triangle.a\_;

std::cout << "2 вершина:\t";

stream >> triangle.b\_;

std::cout << "3 вершина:\t";

stream>> triangle.c\_;

return stream;

}

//тестирование класса Triangle

void tetsTriangle()

{

std::cout << "Тестирование класса Triangle\n";

Triangle triangle1;

Triangle triangle2;

std::cin >> triangle1;

std::cin >> triangle2;

std::cout << triangle1 << triangle2;

triangle1 == triangle2 ? std::cout << "Треугольники совпадают\n" : std::cout << "Треугольники не совпадают\n";

triangle1 < triangle2 ? std::cout << "Площадь треугольника 1 меньше площади треугольника 2\n" : std::cout << "Площадь теругольника 2 больше или равна площади треугольника 1\n";

std::cout << "Введите расстояние, на которое будут перемещены треугольники: ";

std::string value = "";

std::cin >> value;

triangle1 = triangle1 + checkInput(value);

triangle2 += checkInput(value);

std::cout << triangle1 << triangle2;

std::cout << "Конец тестирования класса Triangle\n";

};

1. **Порядок вызова конструкторов и деструкторов**

*Класс Point*

Point (double x, double y)

Point (double x, double y)

Point ()

Point (Point point)

Point (Point point)

~Point (p4)

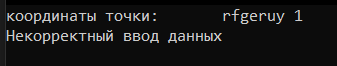
~Point (p5)

~Point (p1)

~Point (p2)

~Point (p3)

# Приложение



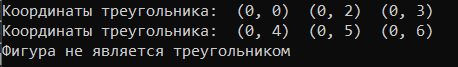
*Рис. 1 Некорректный ввод координат точек*



*Рис. 2 Некорректный ввод расстояния*



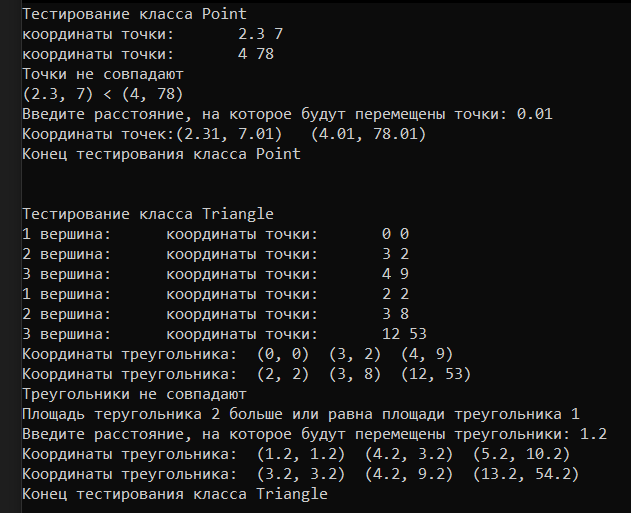
*Рис. 3 Некорректный ввод координат вершин треугольника*



*Рис. 4 Введенные координаты точек не задают треугольник*



*Рис. 5 Некорректный ввод расстояния*



*Рис. 6 Все данные заданы корректно*

# Вывод

В ходе работы:

1. Была изучена и осуществлена перегрузка операторов 2 классов Point (переменные типа Point хранят координаты точек) и Triangle (переменные типа Triangle задают треугольник)
2. Были изучены дружественные функции, с помощью которых была осуществлена перегрузка операторов ввода (>>) и вывода (<<)
3. Была использована конструкция try … catch, которая позволяет эффективнее обрабатывать возникающие в ходе работы программы ошибки