Геометрические фигуры

Индивидуальное задание по ООП C++11

Выполнил

студент гр. 5130904/40003 Д.Р. Никищенко

Руководитель-преподаватель д.к. череповский

Содержание

[Цель работы 3](#_Toc185460287)

[Задачи 3](#_Toc185460288)

[A1: Геометрические фигуры 3](#_Toc185460289)

[A2: Масштабирование фигур 4](#_Toc185460290)

[A3: Составные фигуры 4](#_Toc185460291)

[Детальные требования 5](#_Toc185460292)

[1. Класс Point 5](#_Toc185460293)

[1.1 Поля класса Point: 5](#_Toc185460294)

[1.2 Ошибки и константы класса Point 5](#_Toc185460295)

[1.3 Методы класса Point: 5](#_Toc185460296)

[2. Класс FrameRectangle 7](#_Toc185460297)

[2.1 Поля класса FrameRectangle: 7](#_Toc185460298)

[2.2 Методы класса FrameRectangle: 7](#_Toc185460299)

[3. Класс Shape 8](#_Toc185460300)

[3.1 Методы класса: 9](#_Toc185460301)

[4. Класс Triangle 9](#_Toc185460302)

[4.1 Поля класса Triangle 9](#_Toc185460303)

[4.2 Ошибки и константы класса Triangle 10](#_Toc185460304)

[4.3 Методы класса Triangle 10](#_Toc185460305)

[5. Класс Ring 12](#_Toc185460306)

[5.1 Поля класса Ring 12](#_Toc185460307)

[5.2 Ошибки и константы класса Ring 12](#_Toc185460308)

[5.3 Методы класса Ring 13](#_Toc185460309)

[6. Класс CompositeShape 15](#_Toc185460310)

[6.1 Поля класса CompositeShape 15](#_Toc185460311)

[6.2 Ошибки и константы класса CompositeShape 15](#_Toc185460312)

[6.3 Методы класса CompositeShape 16](#_Toc185460313)

[7. Вспомогательный заголовочный файл Support.h 17](#_Toc185460314)

[7.1 Вспомогательные функции max() и min() 17](#_Toc185460315)

[7.2 Ошибки и константы для main.cpp 17](#_Toc185460316)

[8. Входные данные 18](#_Toc185460317)

[8.1 Программа реализована в виде вложенных меню. 18](#_Toc185460318)

[8.2 Фигура Triangle должна быть задана корректно 23](#_Toc185460319)

[8.3 Фигура Ring должна быть задана корректно 24](#_Toc185460320)

[8.4 Фигура CompositeShape должна быть задан корректно 25](#_Toc185460321)

# Цель работы

Реализовать программу, демонстрирующую свойства классов на языке C++, такие как наследственность и полиморфизм.

# Задачи

## A1: Геометрические фигуры

Все числовые данные в этой работе должны быть значениями с плавающей запятой.

1. Создать класс/структуру Point для представления точки на плоскости с координатами x\_ и y\_

2. Создать класс FrameRectangle, представляющий прямоугольник шириной width\_ и высотой height\_ с центром в точке pos\_. Стороны прямоугольника всегда параллельны осям координат.

3. Создать абстрактный класс Shape, включающий следующие методы:

* getArea() – вычисление площади.
* getFrameRectangle() – получение ограничивающего прямоугольника для

фигуры (объект класса FrameRectangle).

* move() – перемещение центра фигуры, в конкретную точку и в виде смещений по осям абсцисс и ординат.
* scale() - изотропное масштабирование фигуры относительно её центра с

указанным коэффициентом.

* clone() - создание копии фигуры в динамической памяти и получение указателя на эту копию.
* getName() – получение названия фигуры (), метод использовать при выводе.

4. Реализовать два класса – Triangle и Ring - как классы, производные от

класса Shape.

5. Реализовать функцию вывода информации о фигуре.

6. Написать тесты, демонстрирующие работу созданных классов. Демонстрация должна включать полиморфное применение классов.

## A2: Масштабирование фигур

1. Добавить в классы метод scale, осуществляющий изотропное масштабирование фигур относительно их центра с указанным коэффициентом.

2. Написать тесты, проверяющие:

* неизменность ширины и высоты, а также площади фигуры при перемещениях;
* квадратичное изменение площади фигуры при масштабировании;
* наличие и обработку некорректных параметров в методах и конструкторах.

## A3: Составные фигуры

1. Создать производный от супер класса Shape класс CompositeShape, описывающий составную фигуру, состоящую из простых фигур. Необходимо самостоятельно реализовать хранение множества фигур на базе динамического массива. Максимальное количество простых фигур должно задаваться в конструкторе CompositeShape.

2. Реализовать масштабирование и перемещение для CompositeShape.

Масштабирование и перемещение выполнить относительно центра составной фигуры,

за который принимается центр ограничивающего прямоугольника. В реализации можно использовать только методы move и scale без расширения интерфейса Shape.

3. Написать набор тестов, проверяющий корректную работу созданного класса. Для тестирования можно реализовать дополнительные методы.

# Детальные требования

## Класс Point

### 1.1 Поля класса Point:

double x\_ - координата точки, X;

double y\_ - координата точки, X;

### 1.2 Ошибки и константы класса Point

INPUT\_ERROR = "ERROR: invalid input";

### 1.3 Методы класса Point:

Point() – конструктор класса по умолчанию. Присваивает полям x\_ и y\_ значения 0.0;

Point(double x, double y) – конструктор класса. Присваивает полям x\_ и y\_ значения передаваемых переменных типа double x и y соответственно;

Point(const Point& ob) – конструктор копирования. Присваивает полям создаваемого объекта значения полей передаваемого объекта класса Point;

Point(Point&& otherPoint) noexcept – конструктор перемещения. Передает поля передаваемого объекта R-value класса Point создаваемому объекту;

void setX(double x) – метод, передающий значение полю x\_;

void setY(double y) – метод, передающий значение полю y\_;

void move(double k) – метод, передвигающий точку по координатам x и y на значение k;

void move(double kX, double kY) – метод, передвигающий точку по координатам x и y на значение kX и kY соответственно;

double getX() const – метод, возвращающий значение поля x\_;

double getY() const – метод, возвращающий значение поля y\_;

double getDistance(const Point& otherDot) const – метод, вычисляющий расстояние от одной точки до другой

bool operator==(const Point& rightDot) const – оператор, проверяющий равенство двух точек;

bool operator<(const Point& rightDot) const – оператор, проверяющий, находится ли точка слева ближе к началу координат, чем точка справа;

Point operator+(double k) – оператор, прибавляющий полям класса значение k;

Point operator+=(double k) – оператор, прибавляющий полям класса значение k;

Point operator-(double k) – оператор, отнимающий у значений полей класса значение k;

Point operator+(Point& rightDot) – оператор, складывающий точки;

Point operator-(Point& rightDot) – оператор вычитания левой точки от правой;

Point operator\*(double k) – оператор умножения точки на число;

operator double() const – оператор преобразования точки к типу double (вычисление расстояния от начала координат до данной точки);

Point& operator=(const Point& ob) – оператор копирования класса;

Point& operator=(Point&& otherPoint) noexcept – оператор перемещения класса;

friend std::istream& operator>> (std::istream& in, Point& dot) – оператор ввода значений x и y. При некорректном вводе выбрасывает INPUT\_ERROR;

friend std::ostream& operator<< (std::ostream& out, const Point& dot) – оператор вывода значений (x,y);

~Point() = default – деструктор класса (объявлен по умолчанию)

## 2. Класс FrameRectangle

Класс создает прямоугольник. Далее будет использоваться, как рамка для фигуры.

### 2.1 Поля класса FrameRectangle:

double width\_, height\_ - ширина и высота;

Point center\_ - точка, центр прямоугольник;

### 2.2 Методы класса FrameRectangle:

FrameRectangle() – конструктор по умолчанию. Создает вырожденный прямоугольник (точку) с координатами центра (0,0);

FrameRectangle(double width, double height, Point center) – конструктор класса. Создает прямоугольник с шириной width, высотой height и центром в точку center;

FrameRectangle(const FrameRectangle& otherFrameRectangle) – конструктор копирования.;

FrameRectangle(FrameRectangle&& otherFrameRectangle) noexcept – конструктор перемещения.

double getWidth() const – возвращает значение ширины прямоугольника;

double getHeight() const – возвращает значение высоты прямоугольника;

Point getCenter() const – возвращает центр прямоугольника;

Point getLowerLeftPoint() const – возвращает левую нижнюю точку прямоугольника;

Point getHigherRightPoint() const – возвращает правую верхнюю точку прямоугольника;

void setWidth(double width) – присваивает значению ширины значение width;

void setHeight(double height) – присваивает значению высоты значение width;

void setCenter(Point center) – перемещает центр прямоугольника в точку cente;

void swap(FrameRectangle& otherFrameRectangle) noexcept – метод обмена значениями одного объекта класса FrameRectangle с другим;

FrameRectangle& operator=(const FrameRectangle& otherFrame) – оператор копирования;

FrameRectangle& operator=(FrameRectangle&& otherFrame) noexcept – оператор перемещения;

~FrameRectangle() = default – деструктор (по умолчанию);

## 3. Класс Shape

Чисто виртуальный класс-интерфейс для определения методов, общих для производных классов Triangle, Ring и CompositeShape.

### 3.1 Методы класса:

virtual double getArea() const = 0 – виртуальный метод нахождения площади;

virtual FrameRectangle getFrameRectangle() const = 0 – виртуальный метод получения рамки фигуры;

virtual void move(double k) = 0 – виртуальный метод перемещения фигуры на k по x и y;

virtual void move(double k, double p) = 0 – виртуальный метод перемещения фигуры по x и y на k и p соответственно;

virtual void move(Point newCenter) = 0 – виртуальный метод;

virtual void scale(double k) = 0 – виртуальный метод изотропного масштабирования фигуры;

virtual Shape\* clone() const = 0 – виртуальный метод клонирования фигуры;

virtual std::string getName() const = 0 – виртуальный метод получения имени фигуры;

virtual void print(std::ostream& out) const = 0 – виртуальный метод вывода информации о фигуре;

friend std::ostream& operator << (std::ostream& out, Shape& shape) – оператор вывода информации о фигуре;

virtual ~Shape() = default – виртуальный деструктор;

## 4. Класс Triangle

### 4.1 Поля класса Triangle

Point a\_, b\_, c\_, center\_ - вершины треугольника и его центр;

double ab\_, bc\_, ac\_ - стороны треугольника;

### 4.2 Ошибки и константы класса Triangle

TRIANGLE\_NAME = "TRIANGLE";

INVALID\_TRIANGLE = "ERROR: impossible to construct Triangle";

INVALID\_SCALE = "ERROR: too small scale coefficient for triangle";

### 4.3 Методы класса Triangle

Triangle() – конструктор по умолчанию. Создает треугольник с вершинами (0,0), (0,1), (1,0) соответственно;

Triangle(Point a, Point b, Point c) – конструктор класса. Создает треугольник с вершинами в точках a, b, c соответственно;;

Triangle(const Triangle& triangle) – конструктор копирования;

Triangle(Triangle&& triangle) noexcept – конструктор перемещения;

void setA(Point a) – переставляет точку a\_ в a. Если при этом треугольник перестает существовать, выбрасывается ошибка INVALID\_TRIANGLE;

void setB(Point b) – переставляет точку b\_ в b. Если при этом треугольник перестает существовать, выбрасывается ошибка INVALID\_TRIANGLE;

void setC(Point c) – переставляет точку c\_ в c. Если при этом треугольник перестает существовать, выбрасывается ошибка INVALID\_TRIANGLE;

Point getA() const – возвращает вершину a\_;

Point getB() const – возвращает вершину b\_;

Point getC() const – возвращает вершину c\_;

Point getTriangleCenter() const – возвращает center\_;

double getAB() const – возвращает сторону ab\_;

double getBC() const – возвращает сторону bc\_;

double getAC() const – возвращает сторону ac\_;

bool isTriangle() const – проверяет, существует ли треугольник (не лежат ли все точки на одной прямой);

double getPerimeter() const – возвращает периметр треугольника;

void swap(Triangle& triangle) noexcept – метод обмена значениями одного объекта класса Triangle с другим;

virtual double getArea() const override – виртуальный наследуемый метод нахождения площади треугольника;

virtual FrameRectangle getFrameRectangle() const override – виртуальный наследуемый метод нахождения рамки треугольника;

virtual void move(double k) override – виртуальный наследуемый метод перемещения треугольника на k по координатам x и y;

virtual void move(double k, double p) override – виртуальный наследуемый метод перемещения треугольника на k и p по координатам x и y соответственно;

virtual void move(Point newCenter) override – виртуальный наследуемый метод перемещения центра треугольника в точку;

virtual void scale(double k) override – виртуальный наследуемый метод изотропного масштабирования треугольника. Если масштабирования превращает треугольник в вырожденный, выводится ошибка INVALID\_SCALE;

virtual Triangle\* clone() const override – виртуальный наследуемый метод клонирования треугольника;

virtual std::string getName() const override – виртуальный наследуемый метод получения имени треугольника – TRIANGLE\_NAME;

virtual void print(std::ostream& out) const override – виртуальный наследуемый метод вывода информации о треугольнике;

bool operator==(const Triangle& triangle) const – оператор сравнения треугольников;

bool operator<(const Triangle& triangle) const – оператор сравнения площадей треугольников;

void operator+=(double k) – оператор перемещения треугольника на k по

x и y;

Triangle& operator=(const Triangle& triangle) – оператор копирования;

Triangle& operator=(Triangle&& triangle) noexcept – оператор перемещения;

friend std::istream& operator >> (std::istream& in, Triangle& triangle) – оператор ввода. При некорректном вводе трех вершин треугольника выбрасывается ошибка внутри класса Point - INPUT\_ERROR;

friend std::ostream& operator << (std::ostream& out, const Triangle& triangle) – оператор вывода информации о треугольнике;

~Triangle() = default – деструктор (по умолчанию);

## 5. Класс Ring

### 5.1 Поля класса Ring

double radius1\_, radius2\_ - внешний и внутренний радиусы соответственно;

Point center\_ - центр кольца;

### 5.2 Ошибки и константы класса Ring

const std::string RING\_NAME = "RING";

const double PI = 3.141592;

const std::string INPUT\_ERROR = "ERROR: invalid input for Ring";

const std::string RADIUS\_ERROR = "ERROR: radiuses must be positive";

const std::string THIN\_RING\_ERROR = "ERROR: ring is too thin";

const std::string RADIUS\_WARNING = "WARNING: the outer radius of ring should be passed first";

const std::string INVALID\_SCALE = "ERROR: too small scale coefficient for ring";

### 5.3 Методы класса Ring

Ring() – конструктор по умолчанию. Создает кольцо с центром в точке (0,0) и радиусами 2 и 1;

Ring(Point center, double r1, double r2) – конструктор класса. Создает кольцо с центром в точке center и радиусами r1 и r2. Если r1 или r2 отрицательны или близки к нулю, выбрасывается ошибка RADIUS\_WARNING. Если передается слишком тонкое кольцо (радиусы приблизительно равны), выбрасывается ошибка THIN\_RING\_ERROR. Е Если r1 < r2, выводится предупреждение RADIUS\_WARNING, атрибуту radius1\_ передается значение r2,

radius2\_ передаетсяr1;

Ring(const Ring& otherRing) – конструктор копирования;

Ring(Ring&& otherRing) noexcept – конструктор перемещения;

double getOuterRadius() const – возвращает внешний радиус кольца;

double getInnerRadius() const – возвращает внутренний радиус кольца;

Point getCenter() const – возвращает центр кольца;

void setOuterRadius(double r1) – устанавливает внешнему радиусу значений r1. Внутри метода создается временный объект с новыми параметрами, ошибки обрабатываются в конструкторе, в конце значения временного (нового) объекта и старого меняются, временный объект удаляется;

void setInnerRadius(double r2) – устанавливает внутреннему радиусу значений r1. Внутри метода создается временный объект с новыми параметрами, ошибки обрабатываются в конструкторе, в конце значения временного (нового) объекта и старого меняются, временный объект удаляется;

void setCenter(Point newCenter) – перемещает центр кольца в newCenter;

void swap(Ring& newRing) – метод обмена значениями одного объекта класса Ring с другим;

virtual double getArea() const override – виртуальный наследуемый метод нахождения площади кольца;

virtual FrameRectangle getFrameRectangle() const override – виртуальный наследуемый метод нахождения рамки кольца;

virtual void move(double k) override – виртуальный наследуемый метод перемещения кольца на k по координатам x и y;

virtual void move(double k, double p) override – виртуальный наследуемый метод перемещения кольца на k и p по координатам x и y соответственно;

virtual void move(Point newCenter) override – виртуальный наследуемый метод перемещения центра кольца в точку;

virtual void scale(double k) override – виртуальный наследуемый метод изотропного масштабирования кольца. Если масштабирования превращает кольцо в вырожденное, выводится ошибка INVALID\_SCALE;

virtual Ring\* clone() const override – виртуальный наследуемый метод клонирования кольца;

virtual std::string getName() const override – виртуальный наследуемый метод получения имени кольца – RING\_NAME;

virtual void print(std::ostream& out) const override – виртуальный наследуемый метод вывода информации о кольце;

Ring& operator=(const Ring& otherRing) – оператор копирования;

Ring& operator=(Ring&& otherRing) noexcept – оператор перемещения;

friend std::istream& operator >> (std::istream& in, Ring& ring) – оператор ввода кольца. В случае возникновения ошибок в чтении радиусов

выбрасывается ошибка INPUT\_ERROR, если ошибка в чтении точки, выбрасывает оишбка внутри класса Point – INPUT\_ERROR;

friend std::ostream& operator << (std::ostream& out, const Ring& ring) – оператор вывода информации о фигуре;

~Ring() = default – деструктор (по умолчанию);

## 6. Класс CompositeShape

Класс представляет собой динамический массив с заданным максимальным размером, хранящий в себе несколько фигур.

### 6.1 Поля класса CompositeShape

size\_t maxNumberShapes\_{ 0 } – максимальный размер массива; currentNumberShapes\_{ 0 } – количество фигур в массиве;

Shape\*\* shapeArray\_{ nullptr } – массив указателей на фигуры;

### 6.2 Ошибки и константы класса CompositeShape

COMPOSITE\_SHAPE\_NAME = "COMPOSITE SHAPE";

INVALID\_MAX\_NUMBER = "ERROR: max number must be positive integer or 0";

EMPTY\_ARRAY\_ERROR = "ERROR: no shapes in Composite Shape";

MEMORY\_DUMP = "ERROR: not enough memory to allocate";

OUT\_OF\_RANGE\_ERROR = "ERROR: the number of Shapes exceeds the maximum";

### 6.3 Методы класса CompositeShape

CompositeShape() – конструктор по умолчанию. Инициализирует численные поля класса 0, массив – указателем на nullptr;

CompositeShape(int size) – конструктор класса. Создает массив размером size. Если size меньше или равно 0, выбрасывается ошибка INVALID\_MAX\_NUMBER. Если в памяти не хватило места для выделения памяти под массив, выбрасывается ошибка MEMORY\_DUMP;

CompositeShape(const CompositeShape& otherCompShape) – конструктор копирования;

CompositeShape(CompositeShape&& otherCompShape) noexcept – конструктор перемещения;

virtual double getArea() const override – виртуальный наследуемый метод нахождения суммарной площади фигур в списке;

virtual FrameRectangle getFrameRectangle() const override – виртуальный наследуемый метод нахождения рамки композитной фигуры;

virtual void move(double k) override – виртуальный наследуемый метод перемещения композитной фигуры на k по координатам x и y;

virtual void move(double k, double p) override – виртуальный наследуемый метод перемещения композитной фигуры на k и p по координатам x и y соответственно;

virtual void move(Point newCenter) override – виртуальный наследуемый метод перемещения центра композитной фигуры в точку;

virtual void scale(double k) override – виртуальный наследуемый метод изотропного масштабирования композитной фигуры относительно ее центра. Если масштабирования превращает композитную фигуру в вырожденную, выводится ошибка INVALID\_SCALE;

virtual CompositeShape\* clone() const override – виртуальный наследуемый метод клонирования композитной фигуры;

virtual std::string getName() const override – виртуальный наследуемый метод получения имени композитной фигуры – COMPOSITE\_SHAPE\_NAME;

virtual void print(std::ostream& out) const override – виртуальный наследуемый метод вывода информации о составляющих композитной фигуры;

CompositeShape& operator+=(const Shape\* newShape) – оператор добавления фигуры в массив shapeArray\_. Если в массиве недостаточно места выбрасывается ошибка OUT\_OF\_RANGE\_ERROR;

CompositeShape& operator=(const CompositeShape& otherCompShape) – оператор копирования;

CompositeShape& operator=(CompositeShape&& otherCompShape) noexcept – оператор перемещения;

friend std::ostream& operator << (std::ostream& out, CompositeShape& compShape) – оператор вывода информации о составляющих композитной фигуры;

~CompositeShape() – деструктор класса CompositeShape. Удаляет из памяти объекты, хранящиеся в массиве, а затем сам массив.

## 7. Вспомогательный заголовочный файл Support.h

### 7.1 Вспомогательные функции max() и min()

Функции max и min заданы с помощью variadic шаблонов (т.е. принимающих несколько заранее неизвестных по типу данных переменных). Они рекурсивно находят максимумы и минимумы среди нескольких значений соответсвенно.

### 7.2 Ошибки и константы для main.cpp

const std::string INVALID\_SHIFT\_ERROR = "ERROR: shift must be number";

const std::string AMBIGUOS\_SHIFT\_ERROR = "ERROR: shift must be passed as one or two numbers";

const std::string AMBIGUOUS\_TRIANGLE\_ERROR = "ERROR: triangle must be passed as six numbers";

const std::string AMBIGUOUS\_RING\_ERROR = "ERROR: ring must be passed as four numbers";

const std::string AMBIGUOUS\_CENTER\_ERROR = "ERROR: center must be passed as two numbers";

const std::string INVALID\_INPUT = "ERROR: invalid input";

const std::string MOVE\_OPTION\_SHIFT = "k";

const std::string MOVE\_OPTION\_POINT = "point";

## 8. Входные данные

### 8.1 Программа реализована в виде вложенных меню.

#### 8.1.1 Главное меню – выводится сообщение

"Enter the option:

1 - Test Geometric Figures;

2 - Test Figure Scaling;

3 - Test Composite Shape;

Press any other key to quit"

Если вводится число 1 – переходим к тестированию программы для задачи A1, число 2 – A2, число 3 – A3. Если пользователь вводит любое другое значение (буквы, символы, числа с плавающей точкой), программа завершается сообщением “Exiting…”

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

Изображение выглядит как снимок экрана, программное обеспечение, текст, Мультимедийное программное обеспечение

Автоматически созданное описание

Изображение выглядит как компьютер, программное обеспечение, электроника, снимок экрана

Автоматически созданное описание

#### 8.1.2 Тестирование заданий A1-A2: меню

Выводится сообщение

“Test Geometric Figures:

Enter figure option:

t - Triangle,

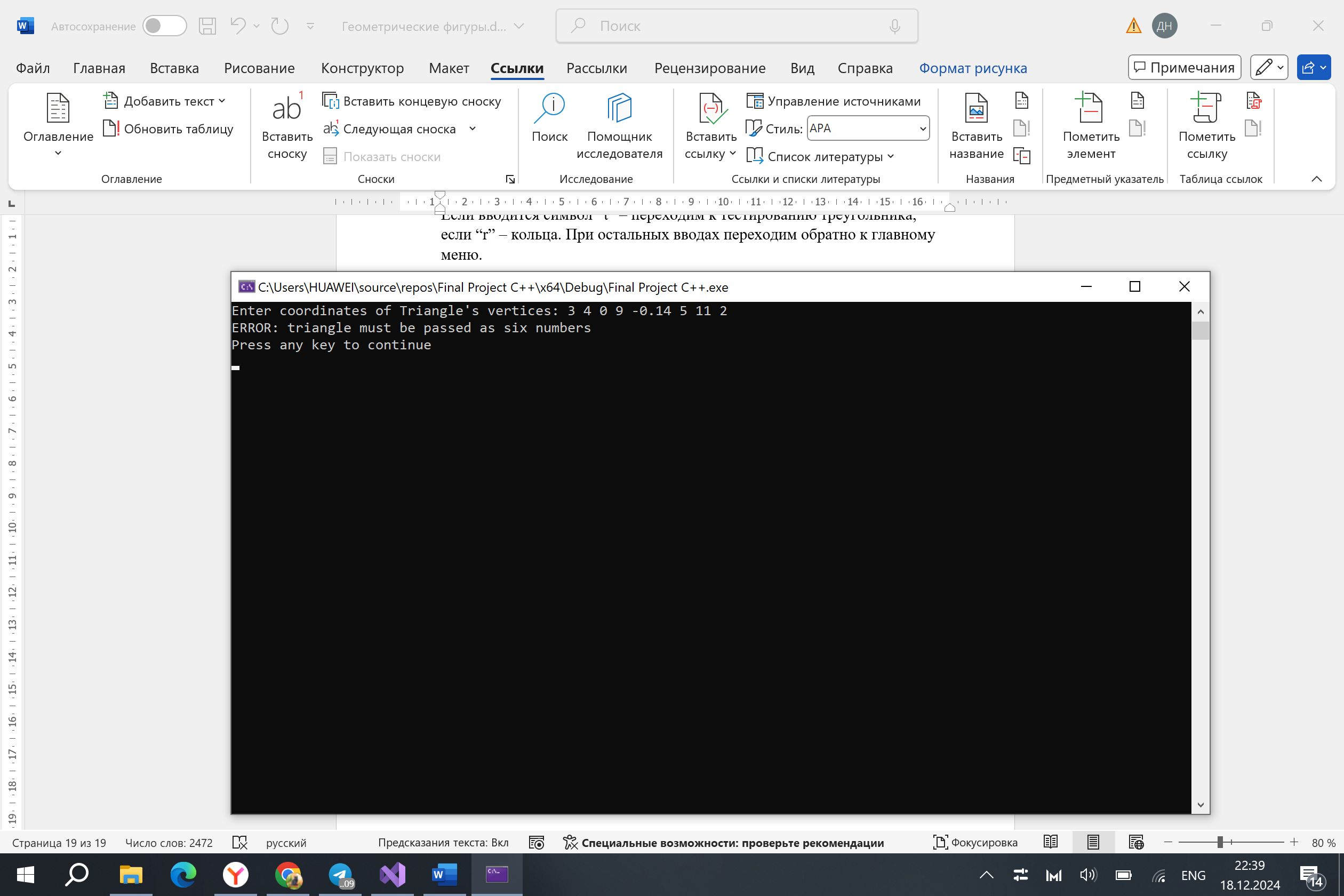
r - Ring;

Press any other key to go back to Test options“

Если вводится символ “t” – переходим к тестированию треугольника, если “r” – кольца. При остальных вводах переходим обратно к главному меню.

#### 8.1.3 Тестирование задания A1-A2: излишние значения Треугольника

Если вводится слишком много точек, выводится сообщение “ERROR: triangle must be passed as six numbers”



Метод move() должен принимать одно/два числа (смещение на число/смещение по числам по координатам). Если вводится больше двух числе, выводится “ERROR: shift must be passed as one or two numbers ”

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

Если вводится слишком много значений для move() в точку, выводится “ERROR: center must be passed as two numbers”

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

#### 8.1.4 Тестирование задания A1-A2: излишние значения Кольца

Если вводится слишком много чисел, выводится сообщение “ERROR: ring must be passed as four numbers”

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Метод move() должен принимать одно/два числа (смещение на число/смещение по числам по координатам). Если вводится больше двух числе, выводится “ERROR: shift must be passed as one or two numbers ”

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

Если вводится слишком много значений для move() в точку, выводится “ERROR: center must be passed as two numbers”

Изображение выглядит как электроника, текст, снимок экрана, программное обеспечение

Автоматически созданное описание

#### 8.1.5 Тестирование задания A3: неправильный ввод числа

Если для числа размера массива передается не число, выводится ошибка

“ERROR: invalid input”

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

### 8.2 Фигура Triangle должна быть задана корректно

#### 8.2.1 Вершины треугольника должны быть числами

Если в конструктор в качестве вершин передаются не числа, выводится сообщение “ERROR: invalid input”

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

#### 8.2.2 Вершины треугольника не должны лежать на одной прямой

Если вершины треугольника лежат на одной прямой, выводится сообщение “ ERROR: impossible to construct Triangle”

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

#### 8.2.3 Коэффицент масштабирования не должен быть близким к нулю

Если коэффициент масштабирования близок к нулю, выводится сообщение “ERROR: too small scale coefficient for triangle”

Изображение выглядит как текст, дисплей, компьютер, снимок экрана

Автоматически созданное описание

### 8.3 Фигура Ring должна быть задана корректно

#### 8.3.1 Центр кольца должен задаваться числами

Если в конструктор в качестве координат центра передаются не числа, выводится сообщение “ERROR: invalid input”

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

#### 8.3.2 Радиусы кольца должны задаваться числами

Если в конструктор в качестве радиусов передаются не числа, выводится сообщение “ERROR: invalid input for Ring”

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

#### 8.3.3 Радиусы кольца должны задаваться числами

Если в конструктор в качестве радиусов передаются числа, близкие к нулю и ниже, выводит сообщение "ERROR: radiuses must be positive"

Изображение выглядит как снимок экрана, текст, программное обеспечение, компьютер

Автоматически созданное описание

#### 8.3.4 Кольцо не должно быть окружностью

Если в качестве радиусов передаются почти одинаковые радиусы (кольцо слишком тонкое), выводится сообщение “ERROR: ring is too thin”

Изображение выглядит как снимок экрана, текст, программное обеспечение, компьютер

Автоматически созданное описание

#### 8.3.5 Кольцо не должно вырождаться

Если коэффициент масштабирования близок к нулю, выводится сообщение “ERROR: too small scale coefficient for ring”

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

### 8.4 Фигура CompositeShape должна быть задан корректно

#### 8.4.1 Фигуры CompositeShape должны быть заданы корректно

(См. пункты 8.1 и 8.2)

#### 8.4.2 Размер массива в CompositesShape должен задаваться числом

В случае ошибки выводится сообщение: “ ERROR: invalid input”

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

#### 8.4.3 Размер массива в CompositesShape должен задаваться натуральным числом

В случае ошибки выводится сообщение: “ ERROR: max number must be positive integer”

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

#### 8.4.4 Под массив должна быть свободная памяти

В случае недостатка памяти отлавливается ошибка bad\_alloc и выводится сообщение "ERROR: not enough memory to allocate"

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

#### 8.4.5 При добавлении элементов в массиве должно быть место

Если в массиве нет места для новой фигуры, выводится сообщение "ERROR: the number of Shapes exceeds the maximum"

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание