МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе№2

по дисциплине «Объектно-ориентированное программирование»

Тема: «Наследование”

Студентка гр. 7304 Юруть Е.А.

Преподаватель Размочаева Н.В.

Санкт-Петербург
2019

Цель работы:

Изучить наследование в C++ при помощи проектирования системы классов для моделирования геометрических фигур (круг, пятиконечная звезда, шестиконечная звезда).

Задача:

Необходимо спроектировать систему классов для моделирования геометрических фигур (в соответствии с полученным индивидуальным заданием). Задание предполагает использование виртуальных функций в иерархии наследования, проектирование и использование абстрактного базового класса. Разработанные классы должны быть наследниками абстрактного класса Shape, содержащего методы для перемещения в указанные координаты, поворота на заданный угол, масштабирования на заданный коэффициент, установки и получения цвета, а также оператор вывода в поток.

Необходимо также обеспечить однозначную идентификацию каждого объекта.

Решение должно содержать:

- условие задания;
- UML диаграмму разработанных классов; • текстовое обоснование проектных решений;
- реализацию классов на языке C++.

Вариант задания:

Вариант 21. Фигуры: квадрат, эллипс, правильный пятиугольник.

Ход работы:

Для хранения координат был создан класс Point.

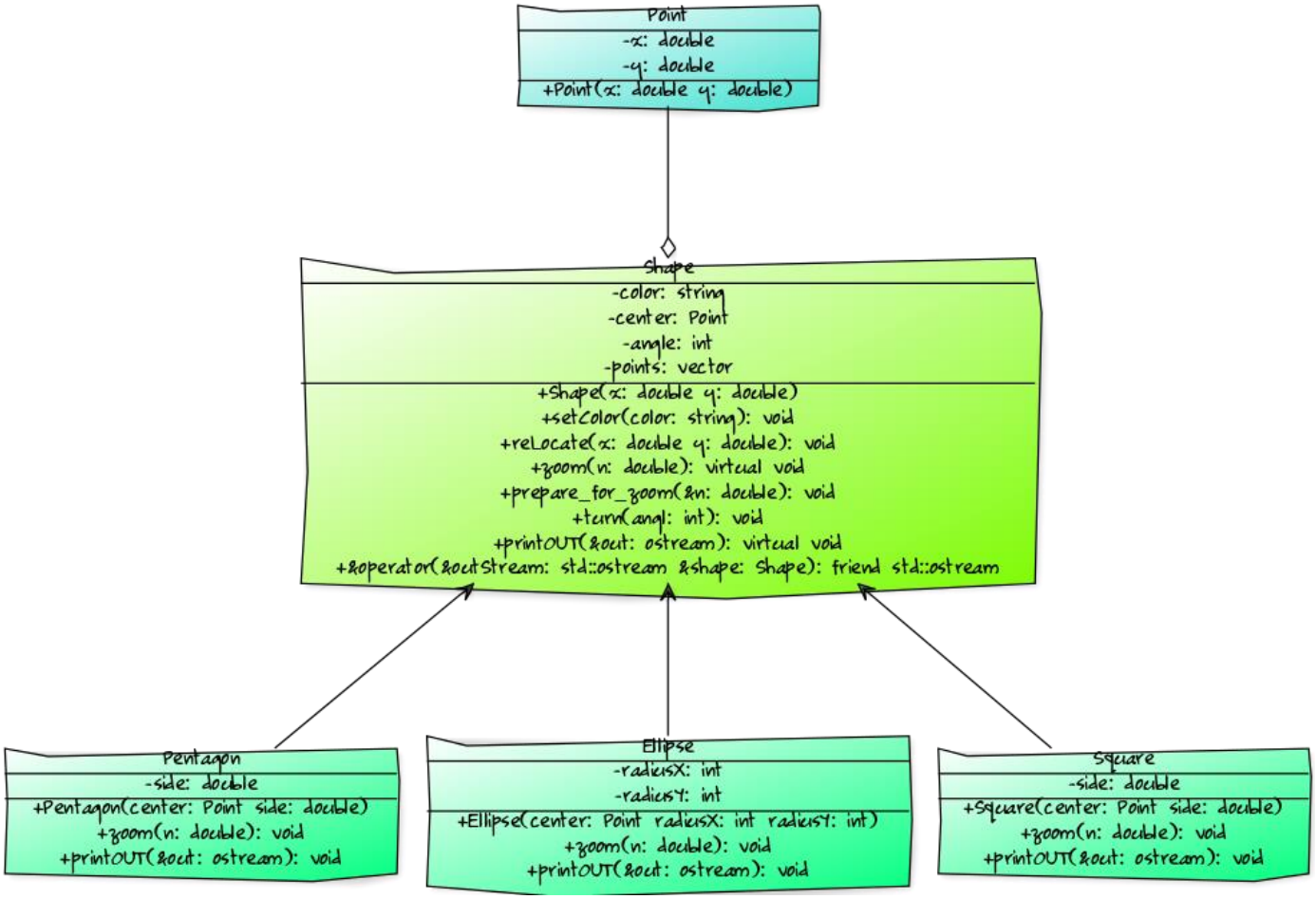
Для последующего создания необходимых геометрических фигур был создан класс Shape, который объединяет все общее для фигур, а именно: координаты центра фигуры, масштаб, угол поворота, цвет. Так же данных классов были написаны общие методы: установка цвета, маштабирование точек, поворот на угол, перемещение в другие координаты, а так же объявлен метод печати данных. Для удобства вывода необходимой информации в консоль без дублирования кода был перегружен оператор “<<” (вывода на консоль)

Класс Square создает квадрат, который определяется в пространстве координатами центра и длинной стороны, с помощью которых определяются четыре точки – вершины квадрата. Также для данного класса написана функции, объявленная в Shape.

Класс Ellipse создает эллипс, который определяется координатами центра и двумя радиусами. Остальные возможности класса аналогичны Square.

Класс Pentagon создает правильный пятиугольник, который определяется координатами центра и длиной стороны. Остальные возможности класса аналогичны Square.

На основе вышеуказанной иерархии была построена UML-диаграмма:



Результаты работы программы:
Квадрат:

```
Индекс фигуры: 0
Создан квадрат
Координаты центра: (0, 0)
Длина стороны квадрата: 2
Точки квадрата:
(-1, 1)
(1, 1)
(1, -1)
(-1, -1)
Цвет: black
Угол: 0
```

Цвет и маштабирование:

```
after zooming +color
Создан квадрат
Координаты центра: (0, 0)
Длина стороны квадрата: 4
Точки квадрата:
(-2, 2)
(2, 2)
(2, -2)
(-2, -2)
Цвет: Green
Угол: 0
```

Поворот:

```
after turning
Создан квадрат
Координаты центра: (0, 0)
Длина стороны квадрата: 4
Точки квадрата:
(-2, -2)
(-2, 2)
(2, 2)
(2, -2)
Цвет: Green
Угол: 90
```

Перемещение:

```
after moving
Создан квадрат
Координаты центра: (3, 3)
Длина стороны квадрата: 6
Точки квадрата:
(1, 1)
(1, 5)
(5, 5)
(5, 1)
Цвет: Green
Угол: 90
```

Для эллипса:

```
Индекс фигуры: 1
Создан эллипс
Координаты центра: (0, 0)
Длины радиусов: 4 , 2
Точки эллипса:
(-4, 0)
(0, 2)
(4, 0)
(0, -2)
Цвет: black
Угол: 0
```

Цвет и маштабирование:

```
after zooming +color
Создан эллипс
Координаты центра: (0, 0)
Длины радиусов: 8 , 4
Точки эллипса:
(-8, 0)
(0, 4)
(8, 0)
(0, -4)
Цвет: Red
Угол: 0
```

Поворот:

```
after turning
Создан эллипс
Координаты центра: (0, 0)
Длины радиусов: 8 , 4
Точки эллипса:
(-4.89859e-16, -8)
(-4, 2.44929e-16)
(4.89859e-16, 8)
(4, -2.44929e-16)
Цвет: Red
Угол: 90
```

Перемещение:

```
after moving
Создан эллипс
Координаты центра: (-2, -2)
Длины радиусов: 8 , 4
Точки эллипса:
(-2, -10)
(-6, -2)
(-2, 6)
(2, -2)
Цвет: Red
Угол: 90
```

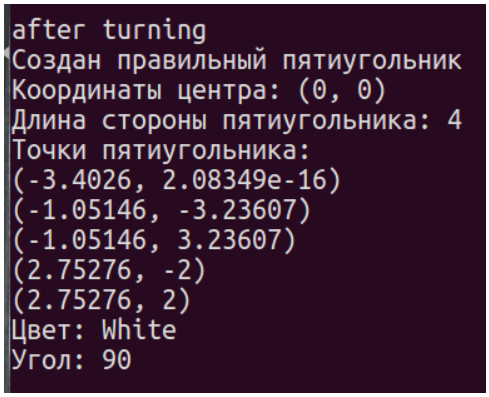
Правильный пятиугольник:

```
Индекс фигуры: 2
Создан правильный пятиугольник
Координаты центра: (0, 0)
Длина стороны пятиугольника: 2
Точки пятиугольника:
(0, 1.7013)
(-1.61803, 0.525731)
(1.61803, 0.525731)
(-1, -1.37638)
(1, -1.37638)
Цвет: black
Угол: 0
```

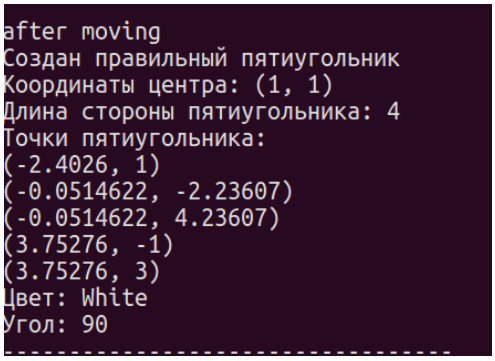
Цвет и маштабирование:

```
after zooming +color
Создан правильный пятиугольник
Координаты центра: (0, 0)
Длина стороны пятиугольника: 4
Точки пятиугольника:
(0, 3.4026)
(-3.23607, 1.05146)
(3.23607, 1.05146)
(-2, -2.75276)
(2, -2.75276)
Цвет: White
Угол: 0
```

Поворот:



Перемещение:



Вывод:

В ходе выполнения лабораторной было изучено наследование в C++, спроектирована система классов для моделирования геометрических фигур (квадрат, эллипс, правильный пятиугольник).

Приложение 1.

Исходный код:

```
  // Квадрат, Эллипс, Правильный пятиугольник
#include <iostream>
#include <string>
#include <vector>
#include <cmath>
#define PI            3.14159265358979323846
using namespace std;
class Point
{
	public:
		double x;
		double y;
		Point(double x, double y) : x(x), y(y) {}
};
class Shape
{
	public:
		Shape(double x, double y) :color("black"), center(x, y), angle(0) {}
		void setColor(string color) // установка цвета
		{
			this->color = color;
		}
		void reLocate(double x, double y) // перемещение в другие координаты (по
центру)
		{
			for (auto& i : points)
			{
				i.x += x - center.x;
				i.y += y - center.y;
			}
			center.x = x;
			center.y = y;
		}
		virtual void zoom(double n)=0;
		void prepare_for_zoom(double& n) // масштабирование каждой точки
		{
			if (n<0)
			{
				n = abs(n);
			}
			for (auto& i : points)
			{
				i.x *= n;
				i.y *= n;
			}
		}
```

```
		void turn(int angl) // поворот на угол
		{
			angl = angl % 360;
			angle += angl;
			double radian = angl*PI / 180;
			for (auto& i : points)
			{
				double x_ = i.x*cos(radian) - i.y*sin(radian);
				double y_ = i.y*cos(radian) + i.x*sin(radian);
				i.x = x_;
				i.y = y_;
			}
		}
		virtual void printOUT(ostream& out) = 0; // печать информации о фигуре
		friend std::ostream& operator<<(std::ostream& outStream, Shape& shape)
		{
			shape.printOUT(outStream);
			return outStream;
		}
	protected:
		string color;
		Point center;
		int angle;
		vector<Point> points;
};
class Square : public Shape
{
	double side;
	public:
		Square(Point center, double side) : Shape(center.x, center.y), side(side)
		{
			points.push_back({ center.x - side / 2, center.y + side / 2 });
			points.push_back({ center.x + side / 2, center.y + side / 2 });
			points.push_back({ center.x + side / 2, center.y - side / 2 });
			points.push_back({ center.x - side / 2, center.y - side / 2 });
		}
		void zoom(double n)
		{
			prepare_for_zoom(n);
			side *= n;
		}
		void printOUT(ostream& out)
		{
			out << "Создан квадрат" << endl;
			out << "Координаты центра: (" << center.x << ", " << center.y << ")" <<
endl;
			out << "Длина стороны квадрата: " << side << endl;
			out << "Точки квадрата:" << endl;
```

```
                for (const auto& i : points)
                {
                        out << "(" << i.x << ", " << i.y << ")\n";
                }
                out << "Цвет: " << color << endl;
                out << "Угол: " << angle << endl;
        }
};
class Ellipse : public Shape
{
        int radiusX, radiusY;
        public:
                Ellipse(Point center, int radiusX, int radiusY) : Shape(center.x, center.y),
radiusX(radiusX), radiusY(radiusY)
                {
                        points.push_back(Point((center.x - radiusX), center.y));
                        points.push_back(Point(center.x, (center.y + radiusY)));
                        points.push_back(Point((center.x + radiusX), center.y));
                        points.push_back(Point(center.x, (center.y - radiusY)));
                //      for(int i=0;i<=360;i++)
                //              points.insert(points.end(),Point(center.x + radiusX *
cos(i*PI/180.0), center.y + radiusY * sin(i*PI/180.0)));
                }
                void zoom(double n)
                {
                        prepare_for_zoom(n);
                        radiusX *= n;
                        radiusY *= n;
                }
                void printOUT(ostream& out)
                {
                        out << "Создан эллипс" << endl;
                        out << "Координаты центра: (" << center.x << ", " << center.y << ")" <<
endl;
                        out << "Длины радиусов: " << radiusX << " , "<< radiusY<< endl;
                        out << "Точки эллипса:" << endl;
                        for (const auto& i : points)
                        {
                                out << "(" << i.x << ", " << i.y << ")\n";
                        }
                        out << "Цвет: " << color << endl;
                        out << "Угол: " << angle << endl;
                }
};
class Pentagon : public Shape
{
        double side;
        public:
```

```
		Pentagon(Point center, double side) : Shape(center.x, center.y), side(side)
		{
			double R = (sqrt(10)*sqrt(5+sqrt(5)))/10)*side;
			points.push_back({ center.x, center.y + R });
			points.push_back({ center.x - sin(72*PI/180.0)*R, center.y +
cos(72*PI/180.0)*R });
			points.push_back({ center.x + sin(72*PI/180.0)*R, center.y +
cos(72*PI/180.0)*R });
			points.push_back({ center.x - sin(36*PI/180.0)*R, center.y -
cos(36*PI/180.0)*R });
			points.push_back({ center.x + sin(36*PI/180.0)*R, center.y -
cos(36*PI/180.0)*R });
		}
		void zoom(double n)
		{
			prepare_for_zoom(n);
			side *= n;
		}
		void printOUT(ostream& out)
		{
			out << "Создан правильный пятиугольник" << endl;
			out << "Координаты центра: (" << center.x << ", " << center.y << ")" <<
endl;
			out << "Длина стороны пятиугольника: " << side << endl;
			out << "Точки пятиугольника:" << endl;
			for (const auto& i : points)
			{
				out << "(" << i.x << ", " << i.y << ")\n";
			}
			out << "Цвет: " << color << endl;
			out << "Угол: " << angle << endl;
		}
};
int main()
{
	Square sq({ 0,0 }, 2);
	cout << sq << endl;
	cout << "" << endl;
	cout << "after zooming +color" << endl;
	sq.zoom(2);
	sq.setColor("Green");
	cout << sq << endl;
	sq.turn(90);
	cout << "" << endl;
	cout << "after turning" << endl;
	cout << sq << endl;
	cout << "" << endl;
	cout << "after moving" << endl;
```

```
        sq.reLocate(3,3);
        cout << sq << endl;
        cout << "/---------------------------------/" << endl;
        Ellipse el({ 0,0 }, 4, 2);
        cout << el;
        cout << "" << endl;
        cout << "after zooming +color" << endl;
        el.setColor("Red");
        el.zoom(2);
        cout << el;
        cout << "" << endl;
        cout << "after turning" << endl;
        el.turn(90);
        cout << el << endl;
        cout << "" << endl;
        cout << "after moving" << endl;
        el.reLocate(-2,-2);
        cout << el << endl;
        cout << "-------------------------------------" << endl;
        Pentagon pen({ 0,0 }, 2);
        cout << pen;
        cout << "" << endl;
        cout << "after zooming +color" << endl;
        pen.setColor("White");
        pen.zoom(2);
        cout << pen;
        cout << "" << endl;
        cout << "after turning" << endl;
        pen.turn(90);
        cout << pen << endl;
        cout << "" << endl;
        cout << "after moving" << endl;
        pen.reLocate(1,1);
        cout << pen;
        cout << "---------------------------------" << endl;
        cout << "/*********************************************/" << endl;
        return 0;
}
```


1