МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Объектно-ориентированное программирование»

Тема: «Наследование»

Студент гр. 7381 _____ Адамов Я.В.

Преподаватель _____ Жангиров Т.М.

Санкт-Петербург

2018

Цель работы.

Ознакомиться с понятиями наследование, полиморфизм, абстрактный класс, изучить виртуальные функции, принцип их работы, способ организации в памяти, раннее и позднее связывания в языке C++.

Задание.

Вариант №1.

Необходимо спроектировать систему классов для моделирования геометрических фигур: квадрат, эллипс, правильный пятиугольник. 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.

Ход работы.

Был реализован класс Color_RGBA_8bit для хранения цвета с альфа-каналом в формате RGBA 8-битной разрядности.

Был реализован абстрактный класс Shape, который хранит сведения о координатах фигуры, угле поворота и цвете (объект класса Color_RGBA_8bit) и имеет методы для перемещения фигуры, поворота и виртуальный метод для масштабирования.

Классы Square, Ellips, Regular_Pentagon наследуются от класса Shape и используются для работы с квадратами, эллипсами и правильными пятиугольниками соответственно.

Была написана небольшая демонстрация работы с классами. Код программы представлен в приложении Б.

UML диаграмма разработанных классов.

UML диаграмма разработанных классов представлена в приложении А.

Вывод.

В ходе выполнения лабораторной работы была спроектирована система классов для работы с геометрическими фигурами. Получены навыки работы с абстрактными классами и виртуальными функциями.

Приложение А. Тестирование.

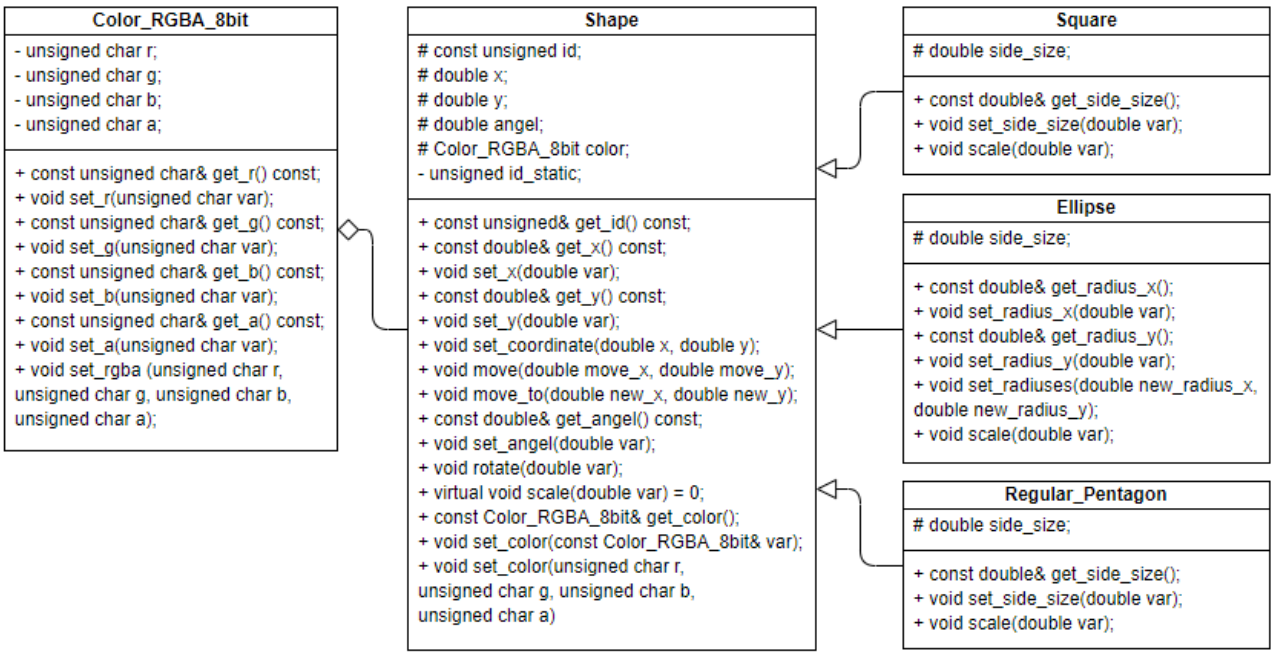

Рисунок 1 – UML диаграмма разработанных классов

Приложение Б. Файл main.cpp.

```
#include <iostream>
#include <cmath>


class Color_RGBA_8bit {
public:
    // Constructors and destructor
    Color_RGBA_8bit(unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a
= 255)
    : r(r), g(g), b(b), a(a) {}

    Color_RGBA_8bit(const Color_RGBA_8bit& var)
    : r(var.r), g(var.g), b(var.b), a(var.a) {}

    // Methods
    const unsigned char& get_r() const noexcept { return r; }
    void set_r(unsigned char var) noexcept { r = var; }

    const unsigned char& get_g() const noexcept { return g; }
    void set_g(unsigned char var) noexcept { g = var; }

    const unsigned char& get_b() const noexcept { return b; }
    void set_b(unsigned char var) noexcept { b = var; }

    const unsigned char& get_a() const noexcept { return a; }
    void set_a(unsigned char var) noexcept { a = var; }

    void set_rgba (unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a
= 0) noexcept {
        this->r = r;
        this->g = g;
        this->b = b;
        this->a = a;
    }

    // Friend functions and classes
    friend std::ostream& operator<<(std::ostream&, const Color_RGBA_8bit&);
    friend class Shape;

    // Operators
    Color_RGBA_8bit& operator = (const Color_RGBA_8bit& var) {
        r = var.r;
        g = var.g;
```

```
        b = var.b;
        a = var.a;
        return *this;
    }

private:
    unsigned char r;
    unsigned char g;
    unsigned char b;
    unsigned char a;
};


std::ostream& operator << (std::ostream& stream, const Color_RGBA_8bit& color) {
    std::cout << "RGBA(" << unsigned(color.r) << ", " << unsigned(color.g) << ", " <<
unsigned(color.b) << ", " << unsigned(color.a) << ")";
    return stream;
}
```

```
class Shape {
public:
    // Constructors and destructor
    Shape(double x = 0, double y = 0, double angel = 0, unsigned char r = 0, unsigned char g = 0,
unsigned char b = 0, unsigned char a = 255)
    : id(id_static), x(x), y(y), color(r, g, b, a) { set_angel(angel); id_static++; }

    Shape(double x, double y, double angel, const Color_RGBA_8bit& color)
    : id(id_static), x(x), y(y), angel(angel), color(color) { set_angel(angel); id_static++; }

    // Methods
    const unsigned& get_id() const noexcept { return id; }

    const double& get_x() const noexcept { return x; }
    void set_x(double var) noexcept { x = var; }

    const double& get_y() const noexcept { return y; }
    void set_y(double var) noexcept { y = var; }

    void set_coordinate(double x, double y) noexcept {
        this->x = x;
```

```
        this->y = y;
    }

    void move(double move_x = 0, double move_y = 0) noexcept {
        x += move_x;
        y += move_y;
    }

    void move_to(double new_x = 0, double new_y = 0) noexcept {
        x = new_x;
        y = new_y;
    }

    const double& get_angel() const noexcept { return angel; }
    void set_angel(double var) noexcept {
        angel = var;
        while (angel < 0)
            angel += 360.0;
        while (angel >= 360.0)
            angel -= 360;
    }

    void rotate(double var) noexcept {
        set_angel(angel + var);
    }

    virtual void scale(double var) = 0;

    // Color
    const Color_RGBA_8bit& get_color() const noexcept { return color; }
    void set_color(const Color_RGBA_8bit& var) noexcept { color = var; }
    void set_color(unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a
= 255) noexcept { color.set_rgba(r, g, b, a); }
    const unsigned char& get_r() const noexcept { return color.r; }
    void set_r(unsigned char var) noexcept { color.r = var; }
    const unsigned char& get_g() const noexcept { return color.g; }
    void set_g(unsigned char var) noexcept { color.g = var; }
    const unsigned char& get_b() const noexcept { return color.b; }
    void set_b(unsigned char var) noexcept { color.b = var; }
    const unsigned char& get_a() const noexcept { return color.a; }
    void set_a(unsigned char var) noexcept { color.a = var; }

protected:
    const unsigned id;
    double x;
```

```
    double y;
    double angel;
    Color_RGBA_8bit color;


private:
    static unsigned id_static;

};


unsigned Shape::id_static = 0;
```

```
class Square : public Shape {
public:
    // Constructors and destructor
    Square(double side_size = 0, double x = 0, double y = 0, double angel = 0, unsigned char r =
0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 255)
    : side_size(side_size < 0 ? -side_size : side_size), Shape(x, y, angel, r, g, b, a) {}

    Square(double side_size, double x, double y, double angel, const Color_RGBA_8bit color)
    : side_size(side_size < 0 ? -side_size : side_size), Shape(x, y, angel, color) {}

    Square(const Square& var)
    : side_size(var.side_size), Shape(var.x, var.y, var.angel, var.color) {}

    // Methods
    const double& get_side_size() const noexcept { return side_size; }
    void set_side_size(double var) noexcept { side_size = var < 0 ? -var : var; }

    void scale(double var) noexcept override {
        side_size *= var < 0 ? -var : var;
    }

    // Friend functions
    friend std::ostream& operator<<(std::ostream&, const Square&);
```

```
protected:
    double side_size;
};


std::ostream& operator << (std::ostream& stream, const Square& square) {
    std::cout << "Square[id: " << square.id << ", side size: " << square.side_size << ",
coordinate(" << square.x << ", " << square.y << "), angel: " << square.angel << ", " <<
square.color << ']';
    return stream;
}
```

```
class Ellipse : public Shape {
public:
    // Constructors and destructor
    Ellipse(double radius_x = 0, double radius_y = 0, double x = 0, double y = 0, double angel =
0, unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 255)
    : radius_x(radius_x < 0 ? -radius_x : radius_x), radius_y(radius_y < 0 ? -radius_y :
radius_y), Shape(x, y, angel, r, g, b, a) {}

    Ellipse(double radius_x, double radius_y, double x, double y, double angel, const
Color_RGBA_8bit& color)
    : radius_x(radius_x < 0 ? -radius_x : radius_x), radius_y(radius_y < 0 ? -radius_y :
radius_y), Shape(x, y, angel, color) {}

    Ellipse(const Ellipse& var)
    : radius_x(var.radius_x), radius_y(var.radius_y), Shape(var.x, var.y, var.angel, var.color) {}

    // Methods
    const double& get_radius_x() const noexcept { return radius_x; }
    void set_radius_x(double var) noexcept { radius_x = var < 0 ? -var : var; }

    const double& get_radius_y() const noexcept { return radius_y; }
    void set_radius_y(double var) noexcept { radius_y = var < 0 ? -var : var; }
```

```
    void set_radiuses(double new_radius_x, double new_radius_y){
        set_radius_x(new_radius_x);
        set_radius_y(new_radius_y);
    }

    void scale(double var) noexcept override {
        double temp = var < 0 ? -var : var;
        radius_x *= temp;
        radius_y *= temp;
    }

    // Friend functions
    friend std::ostream& operator<<(std::ostream&, const Ellipse&);

protected:
    double radius_x;
    double radius_y;
};


std::ostream& operator << (std::ostream& stream, const Ellipse& ellipse) {
    std::cout << "Ellipse[id: " << ellipse.id << ", radius x: " << ellipse.radius_x << ", radius
y: " << ellipse.radius_y << ", coordinate(" << ellipse.x << ", " << ellipse.y << "), angel: " <<
ellipse.angel << ", " << ellipse.color << ']';
    return stream;
}
```

```
class Regular_Pentagon : public Shape {
public:
    // Constructors and destructor
    Regular_Pentagon(double side_size = 0, double x = 0, double y = 0, double angel = 0, unsigned
char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 255)
    : side_size(side_size < 0 ? -side_size : side_size), Shape(x, y, angel, r, g, b, a) {}

    Regular_Pentagon(double side_size, double x, double y, double angel, const Color_RGBA_8bit&
color)
```

```
    : side_size(side_size < 0 ? -side_size : side_size), Shape(x, y, angel, color) {}

    Regular_Pentagon(const Regular_Pentagon& var)
    : side_size(var.side_size), Shape(var.x, var.y, var.angel, var.color) {}

    // Methods
    const double& get_side_size() const noexcept { return side_size; }
    void set_side_size(double var) noexcept { side_size = var < 0 ? -var : var; }

    void scale(double var) noexcept override {
        side_size *= var < 0 ? -var : var;
    }

    // Friend functions
    friend std::ostream& operator<<(std::ostream&, const Regular_Pentagon&);

protected:
    double side_size;
};


std::ostream& operator << (std::ostream& stream, const Regular_Pentagon& pentagon) {
    std::cout << "Regular_Pentagon[id: " << pentagon.id << ", side size: " << pentagon.side_size
<< ", coordinate(" << pentagon.x << ", " << pentagon.y << "), angel: " << pentagon.angel << ", "
<< pentagon.color << ']';
    return stream;
}
```

```
int main() {
    std::cout << "Demonstration.\n" << std::endl;

    Color_RGBA_8bit color_1(45, 15, 115);
    std::cout << color_1 << "\n" << std::endl;

    Square square_1(5, 6.24, -56.11, 14, color_1);
    std::cout << square_1 << std::endl;
```

```
    Square square_2(square_1);
    square_2.move(415.15, -100);
    square_2.rotate(600.15);
    square_2.scale(1.5);
    std::cout << square_2 << std::endl;

    std::cout << std::endl;

    Ellipse ellipse_1(25, 50, -100, -150.15, 0);
    std::cout << ellipse_1 << std::endl;

    std::cout << std::endl;

    Regular_Pentagon pentagon_1(25, 15, 15, 25.234, 54, 23, 14);
    pentagon_1.move_to(-pentagon_1.get_x(), -pentagon_1.get_y());
    pentagon_1.set_angel(0);
    std::cout << pentagon_1 << std::endl;
    pentagon_1.set_r(70);
    std::cout << pentagon_1.get_color() << std::endl;


    return 0;
}
```