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

Студент гр. 7304 _____ Шарапенков И.И.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2019

Цель работы.

Изучить механизм наследования в языке C++, научиться проектировать систему классов.

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 20. Прямоугольник, параллелограмм, шестиконечная звезда.

Описание реализации.

1. Для хранения цвета был создан отдельный класс **Color**. Таким способом была получена абстракция над цветовой моделью RGB. Для удобного вывода цветовых координат был перегружен оператор вывода <<. Таким образом, в дальнейшем можно не заботиться об отдельной обработке цветовых координат при печати в консоль.
2. Аналогично был реализован класс **Coord**, абстрагирующий понятие координат точки в декартовой системе.
3. Для дальнейшего создания классов геометрических фигур был создан абстрактный базовый класс **Shape**, который определяет общие для всех геометрических форм данные и методы.

   К общим данным относятся: *color* – цвет фигуры, *position* – позиция фигуры в декартовой системе координат, *angle* – угол поворота фигуры относительно *position* и начального положения (начальное положение определяется конкретной фигурой), *id* – идентификатор объекта.

   К общим методам с общей реализацией относятся: *move, rotate, setColor, getColor*. *Move* и *rotate* используют только координаты фигуры в пространстве и угол поворота, а так как данные эти данные содержатся в базовом классе, то и методы можно реализовать общие для всех потомков. Аналогично и для *setColor, getColor*.

   Также был перегружен оператор вывода в консоль. Это было сделано для того, чтобы общие для всех фигур данные можно было вывести в одной функцией базового класса, не дублируя код.

   Наконец, метод *scale* был объявлен как чистая виртуальная функция. Причина этого в том, что базовый класс **Shape** не содержит данные, определяющие размер и форму фигуры, поэтому нельзя описать алгоритм её масштабирования.

   Поле *id* представляет собой уникальный идентификатор объекта, чтобы при совпадении всех прочих параметров можно было определить

какой объект перед нами. Его реализация заключается в использовании глобальной переменной – счетчика.

Так как у нас уже реализован механизм определения позиции и поворота фигуры в пространстве, в дальнейшем можно абстрагироваться от системы координат. То есть перейти в любую удобную систему координат конкретной фигуры и не задумываться о зависимости ее параметров от исходной системы координат.

4. Класс **Rectangle** реализует класс прямоугольника. Каждый прямоугольник можно однозначно описать шириной и высотой. Действует следующее соглашение, что высота – *y* координата, ширина – *x*. (Если за центр координат взять центр фигуры). Также был перегружен оператор вывода для вывода информации о фигуре. Метод *scale* для этого класса представляет собой умножение ширины и высоты на коэффициент масштабирования.
5. Класс **Hexagram** реализует шестиконечную звезду. Определяется эта фигура расстоянием между двумя противоположными лучами звезды (высота). При этом в начальном положении луч направлен вверх. Остальное аналогично классу **Rectangle.** Метод *scale* для этого класса представляет собой умножение высоты на коэффициент масштабирования.
6. Класс **Parallelogram** реализует параллелограмм. Можно было бы его наследовать от **Rectangle** или наоборот, но было решено создать его как отдельный класс. Реализация такая же как у **Rectangle,** но есть дополнительное поле *wh_angle,* которое определяет угол между «шириной» и «высотой» для верхнего левого угла параллелограмма. Таким образом, по этим данным можно однозначно построить параллелограмм.
7. На основе иерархии классов была построена UML диаграмма, представленная на рисунке 1.

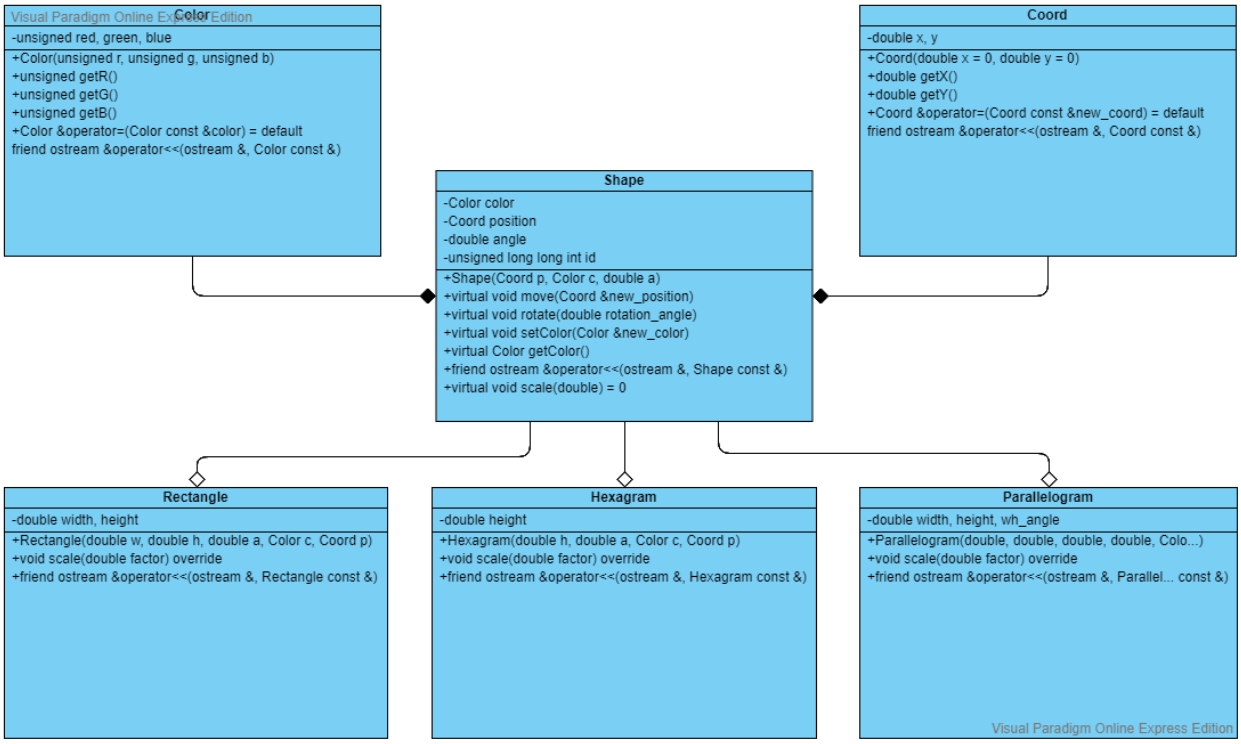

Рис 1. UML диаграмма разработанных классов

Тестирование.

```
int main() {
    Hexagram hex(120);
    Rectangle rect(120, 130);
    Parallelogram para(10, 20);

    cout << hex << endl;
    cout << rect << endl;
    cout << para << endl;

    hex.move(Coord(20, 30));
    rect.rotate(100);
    para.scale(1.2);
    rect.setColor(Color(100, 200, 30));

    cout << rect.getColor() << endl << hex.getColor() << endl;

    cout << hex << endl;
    cout << rect << endl;
    cout << para << endl;

    return 0;
}
```

Консольный вывод после выполнения данного кода:

```
Shape ID: 0
Color: (255,255,255)
Position of the center: (0,0)
Clockwise rotation angle: 0
Shape type: Hexagram
The distance between opposing rays: 120

Shape ID: 1
Color: (255,255,255)
Position of the center: (0,0)
Clockwise rotation angle: 0
Shape type: Rectangle
Width: 120
Height: 130

Shape ID: 2
Color: (255,255,255)
Position of the center: (0,0)
Clockwise rotation angle: 0
Shape type: Parallelogram
Width: 10
Height: 20
Angle between width and height: 3.14159

(100,200,30)
(255,255,255)

Shape ID: 0
Color: (255,255,255)
Position of the center: (20,30)
Clockwise rotation angle: 0
Shape type: Hexagram
The distance between opposing rays: 120

Shape ID: 1
Color: (100,200,30)
Position of the center: (0,0)
Clockwise rotation angle: 100
Shape type: Rectangle
Width: 120
Height: 130

Shape ID: 2
Color: (255,255,255)
Position of the center: (0,0)
Clockwise rotation angle: 0
Shape type: Parallelogram
Width: 12
Height: 24
Angle between width and height: 3.14159
```

Вывод.

В ходе выполнения данной лабораторной работы был изучен механизм наследования в языке C++, была спроектирована и реализована система классов, описывающая иерархию геометрических фигур.