

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

ОТЧЕТ
по лабораторной работе №2
по дисциплине «ООП»
Тема: Наследование

Студент гр. 7304	_____	Абдульманов Э.М
Преподаватель	_____	Размочаева Н.В

г. Санкт-Петербург
2019

Цель работы

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 так же обеспечить однозначную идентификацию каждого объекта.

Решение должно содержать:

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

Ход работы

1. Были созданы вспомогательные структуры данных
 - a. `Struct Color` – структура цвета, содержит три поля, которые хранят числа от 0 до 255 и характеризуют RGB.
 - b. `Class Point` – класс точки, которое описывает координаты точки по оси X и Y.
2. Был создан абстрактный класс `Shape`, который содержит такие поля как: Цвет фигуры, точка центра фигуры, однозначный идентификатор, и вектор множества точек данной фигуры. И следующие методы:
 - a. `void setColor(Color color)` – устанавливает цвет фигуры
 - b. `const Color getColor() const` – выдает цвет фигуры
 - c. `const Point& getCenter() const` – выдает центр фигуры
 - d. `void Moving(Point newCenter)` – перемещает фигуру в заданную точку
 - e. `void Rotate(int angle,int direction)` – поворачивает фигуру на определенный угол либо по часовой, либо против часовой стрелки
 - f. `virtual void Scale(double coefficient)` – виртуальная функция масштабирования фигуры на заданный коэффициент, которая будет переопределяться в классах наследниках
 - g. `friend ostream& operator<<(ostream& out,const Shape& shape)` – переопределения метода вывода фигуры на экран
3. Был создан класс `Square`, который наследовался от класса `Shape`. Он имеет дополнительное поле – `length` (длину стороны квадрата) и переопределяет базовый метод `Scale(int angle,int direction)`,

- масштабируя длину квадрата. А так же метод `friend ostream& operator<<(ostream& out,const Square& square)`
4. Был создан класс `Ellipse`, который наследовался от класса `Shape`. Он имеет два дополнительных поля: `smallRadius` и `bigRadius` и переопределяет базовый метод `Scale(int angle,int direction)`, масштабируя длину маленького радиуса и большого радиуса. А так же метод `friend ostream& operator<<(ostream& out,const Ellipse& ellipse)`
 5. Был создан класс `RegularPentagon`, который наследовался от класса `Shape`. Он имеет дополнительное поле – `length` (длину стороны правильного пятиугольника) и переопределяет базовый метод `Scale(int angle,int direction)`, масштабируя длину пятиугольника. А так же метод `friend ostream& operator<<(ostream& out,const RegularPentagon & regulatPentagon)`.

Обоснование

В данной работе в абстрактном классе `Shape` только один метод `Scale` является виртуальным. Это происходит из-за того, что любую фигуру можно представить в виде множества точек. Виртуальными методами не являются методы `Rotate` и `Moving`, потому что:

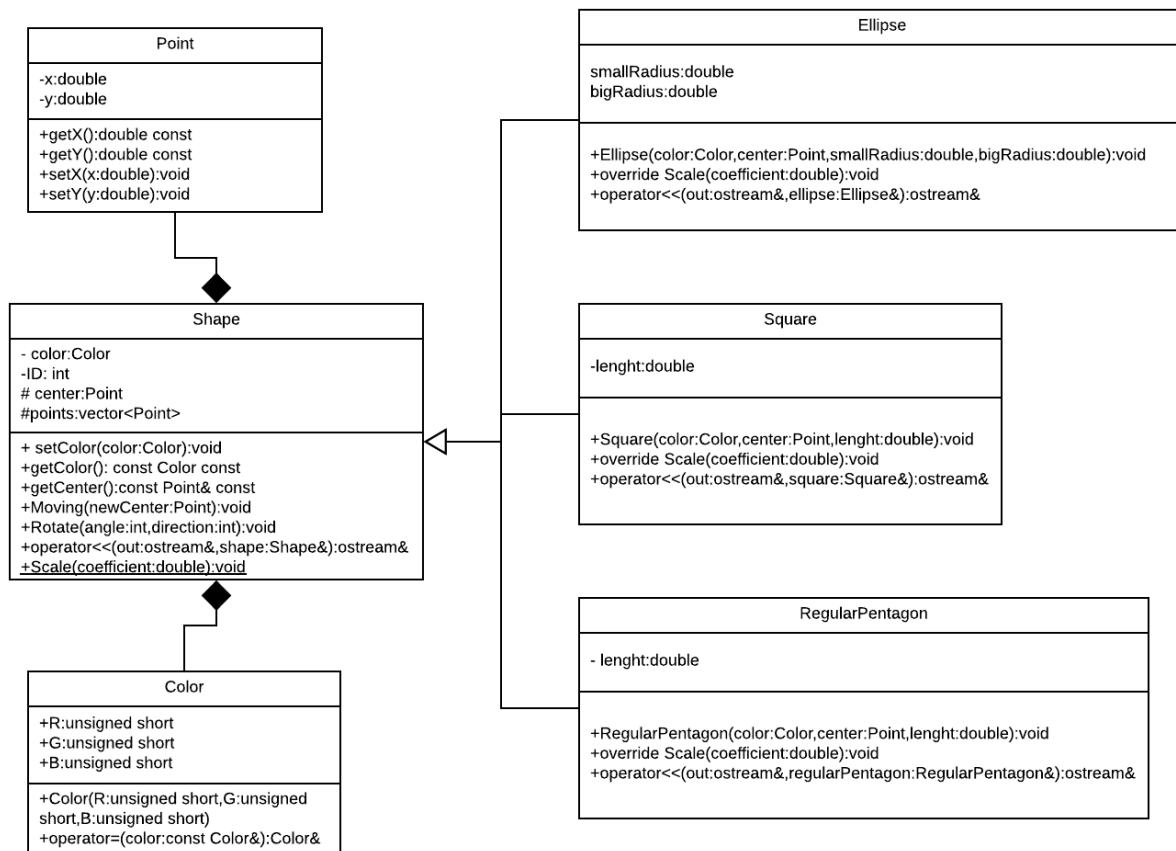
- `Rotate` можно представить как умножение каждой точки на матрицу поворота и тем самым получение новых координат. В итоге изменяется только поле `points(vector<Point> points)`, которое находится в абстрактном классе `Shape`. Вообще изменяются только координаты самой фигуры.
 - `Moving` это перемещение каждой точки фигуры на определенное расстояние. Для любой фигуры можно найти вектор, который будет показывать направление точки относительно центра. Следовательно мы можем просто перенести центр в заданную точку и переместить каждую точку в соответствие с новым центром фигуры.

Переопределение в классах наследниках метода `Scale` требуется потому что, каждая фигура обладает своими индивидуальными свойствами. В данном случае, квадрат и правильный пятиугольник имеет длину стороны, а Эллипс имеет два радиуса(большой и меньший). Поэтому масштабирование должно изменять атрибуты, которые не являются полями абстрактного класса `Shape`.

Поля цвет, координаты центра и вектор точек фигуры являются общими для любой фигуры, поэтому они содержатся в абстрактном классе `Shape`. Конструкторы классов наследников всегда вызывают конструктор класса `Shape` и плюс для каждой отдельной фигуры, определенным образом вычисляются точки это фигуры. Так же, квадрат можно описать 4 точками и центром, правильный пятиугольник – 5 точками и центром, эллипс – 360 точек и центр. Для вычисления каждой

точки эллипса используется соответствующая формула, которая по радиусам и углу определяет положение точки относительно центра.

UML диаграмма классов



Примеры работы программы

- Пример (Квадрат):

- Входные данные

```
Square square(Color(255,10,234),Point(50,50),10);
cout<<square<<endl;
square.Moving(Point(100,100));
cout<<square<<endl;
square.Scale(1.1);
cout<<square<<endl;
square.Rotate(90,1);
cout<<square<<endl;
```

- Выходные данные

```
ID shape=1221
Color`s shape:255 10 234
Center`s shape:(50,50)
Left-Up point`s square:(45,45)
Lenght`s square:10
```

```
ID shape=1221
Color`s shape:255 10 234
Center`s shape:(100,100)
Left-Up point`s square:(95,95)
Lenght`s square:10
```

```
ID shape=1221
Color`s shape:255 10 234
Center`s shape:(100,100)
Left-Up point`s square:(94.5,94.5)
Lenght`s square:11
```

```
ID shape=1221
Color`s shape:255 10 234
Center`s shape:(100,100)
Left-Up point`s square:(105.5,94.5)
Lenght`s square:11
```

- Пример 2 (Правильный пятиугольник)

- Входные данные

```
RegularPentagon regularPentagon(Color(56,243,51),Point(70,70),20);
cout<<regularPentagon<<endl;
regularPentagon.Moving(Point(100,100));
cout<<regularPentagon<<endl;
regularPentagon.Scale(1.1);
cout<<regularPentagon<<endl;
regularPentagon.Rotate(90,1);
cout<<regularPentagon<<endl;
```

- Выходные данные

```
ID shape=2684
Color`s shape:56 243 51
Center`s shape:(70,70)
point`s RegularPentagon:
(101.623,113.525)
(38.3772,113.525)
(18.8333,53.3749)
(70,16.2001)
(121.167,53.3749)
Lenght`s RegularPentagon:20
```

```
ID shape=2684
Color`s shape:56 243 51
Center`s shape:(100,100)
point`s RegularPentagon:
    (131.623,143.525)
    (68.3772,143.525)
    (48.8333,83.3749)
    (100,46.2001)
    (151.167,83.3749)
Lenght`s RegularPentagon:20
```

```
ID shape=2684
Color`s shape:56 243 51
Center`s shape:(100,100)
point`s RegularPentagon:
    (134.785,147.878)
    (65.2149,147.878)
    (43.7166,81.7124)
    (100,40.8201)
    (156.283,81.7124)
Lenght`s RegularPentagon:69
```

```
ID shape=2684
Color`s shape:56 243 51
Center`s shape:(100,100)
point`s RegularPentagon:
    (52.1225,134.785)
    (52.1225,65.2149)
    (118.288,43.7166)
    (159.18,100)
    (118.288,156.283)
Lenght`s RegularPentagon:69
```

- Пример 3(Эллипс)

- Входные данные

```
Ellipse ellipse(Color(21,325,213),Point(100,100),20,40);
cout<<ellipse<<endl;
ellipse.Moving(Point(110,110));
cout<<ellipse<<endl;
ellipse.Scale(1.1);
cout<<ellipse<<endl;
ellipse.Rotate(90,1);
cout<<ellipse<<endl;
```

- Выходные данные

```
ID shape=4160
Color`s shape:21 69 213
Center`s shape:(100,100)
point`s Left:(140,100)
BigRadius:40
SmallRadius:20

ID shape=4160
Color`s shape:21 69 213
Center`s shape:(110,110)
point`s Left:(150,110)
BigRadius:40
SmallRadius:20

ID shape=4160
Color`s shape:21 69 213
Center`s shape:(110,110)
point`s Left:(154,110)
BigRadius:44
SmallRadius:22

ID shape=4160
Color`s shape:21 69 213
Center`s shape:(110,110)
point`s Left:(110,154)
BigRadius:44
SmallRadius:22
```

Вывод

В ходе данной работы было изучено наследование классов в c++. Была спроектирована система классов для моделирования геометрических фигур. Были использованы виртуальных функций в иерархии наследования. Разработанные классы являются 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.