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. 7303 _____ Алексо А.А.

Преподаватель _____ Размочаева Н.В.

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 точке 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 переопределятся 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 переопределяетбазовыйметодScale(intangle,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 переопределяетбазовыйметодScale(intangle,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

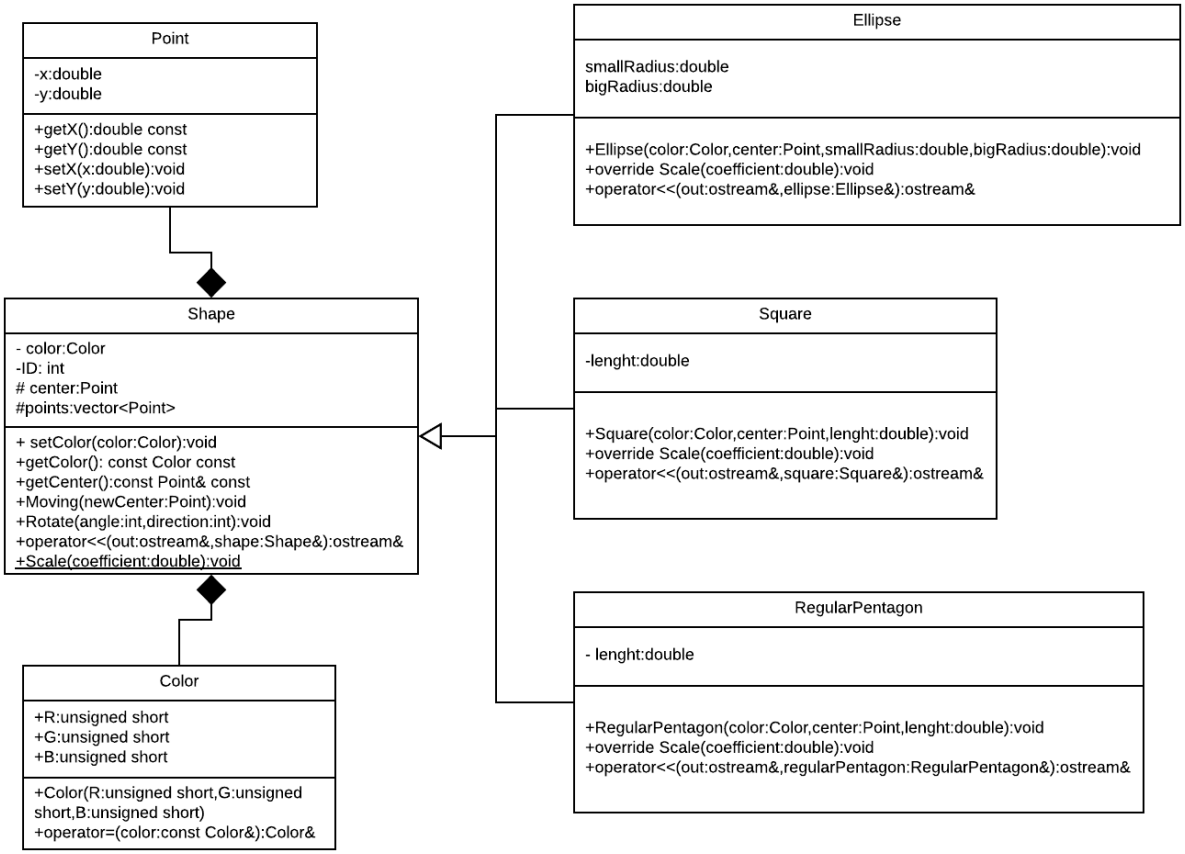
В данной работе в абстрактном классе Shape только один метод Scale является виртуальным. Это происходит из-за того, что любую фигуру можно представить в виде множества точек. Виртуальными методами не являются методы Rotate и Moving, потому что:

- 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. Вообщем 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(больший 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

```
cout << square << endl;
square * 0.5;
cout << square << endl;
square * 1;
cout << square << endl;
square * 3;
cout << square << endl;
square + -5;
cout << square << endl;
square.Scale(0.5);
cout << square << endl;
square.Move(Point(100, 100));
cout << square << endl;
```

  - Выходные данные

```
Shape ID = 1
Color`s shape:255 10 144
Center`s shape:(50,50)
Left-Up point`s square:(42.5,42.5)
Lenght`s square:15

* 0.5
Shape ID = 1
Color`s shape:255 10 144
Center`s shape:(50,50)
Left-Up point`s square:(46.25,46.25)
Lenght`s square:7.5

* 1
Shape ID = 1
Color`s shape:255 10 144
Center`s shape:(50,50)
Left-Up point`s square:(46.25,46.25)
Lenght`s square:7.5

* 3
Shape ID = 1
Color`s shape:255 10 144
Center`s shape:(50,50)
Left-Up point`s square:(38.75,38.75)
Lenght`s square:22.5

+ -5
Shape ID = 1
Color`s shape:255 10 144
Center`s shape:(45,45)
Left-Up point`s square:(33.75,33.75)
Lenght`s square:22.5

Shape ID = 1
Color`s shape:255 10 144
Center`s shape:(45,45)
Left-Up point`s square:(39.375,39.375)
Lenght`s square:11.25
```

- Пример 2(Эллипс)
  - Входные данные

```
cout << ellipse << endl;
ellipse * 2;
cout << ellipse << endl;
ellipse.Scale(0.5);
cout << ellipse << endl;
ellipse.Scale(2);
cout << ellipse << endl;
ellipse + 15;
cout << ellipse << endl;
```

- Выходные данные

```
Shape ID = 2
Color`s shape:255 10 144
Center`s shape:(50,50)
point`s Left:(70,50)
BigRadius:20
SmallRadius:15

* 2
Shape ID = 2
Color`s shape:255 10 144
Center`s shape:(50,50)
point`s Left:(90,50)
BigRadius:40
SmallRadius:30

Shape ID = 2
Color`s shape:255 10 144
Center`s shape:(50,50)
point`s Left:(70,50)
BigRadius:20
SmallRadius:15

Shape ID = 2
Color`s shape:255 10 144
Center`s shape:(50,50)
point`s Left:(90,50)
BigRadius:40
SmallRadius:30

+ 15
Shape ID = 2
Color`s shape:255 10 144
Center`s shape:(65,65)
point`s Left:(105,65)
BigRadius:40
SmallRadius:30
```

- Пример 3(Правильный пятиугольник)
  - Входные данные

```
cout << regularrentagon << endl;
regularrentagon * 2;
cout << regularrentagon << endl;
regularrentagon * 1;
cout << regularrentagon << endl;
regularrentagon * 3;
cout << regularrentagon << endl;
regularrentagon + 10;
cout << regularrentagon << endl;
regularrentagon.Scale(0.5);
cout << regularrentagon << endl;
regularrentagon.Move(Point(100, 100));
cout << regularrentagon << endl;
```

- Выходные данные

```
* 2
Shape ID = 3
Color`s shape:255 10 144
Center`s shape:(50,50)
point`s RegularPentagon:
   (97.4654,115.265)
   (2.63865,115.34)
   (-26.7937,25.1969)
   (49.8072,-30.6996)
   (126.674,24.8302)
Length`s RegularPentagon:94

* 1
Shape ID = 3
Color`s shape:255 10 144
Center`s shape:(50,50)
point`s RegularPentagon:
   (97.4654,115.265)
   (2.63865,115.34)
   (-26.7937,25.1969)
   (49.8072,-30.6996)
   (126.674,24.8302)
Length`s RegularPentagon:94

* 3
Shape ID = 3
Color`s shape:255 10 144
Center`s shape:(50,50)
point`s RegularPentagon:
   (192.396,245.795)
   (-92.084,246.021)
   (-180.381,-24.4094)
   (49.4216,-192.099)
   (280.023,-25.5093)
Length`s RegularPentagon:284

+ 10
Shape ID = 3
Color`s shape:255 10 144
Center`s shape:(60,60)
point`s RegularPentagon:
   (202.396,255.795)
   (-82.084,256.021)
   (-170.381,-14.4094)
   (59.4216,-182.099)
```

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.