**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. 7382 |  | Находько А.Ю. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2019

**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++. В соответствии с индивидуальным заданием разработать систему классов для представления геометрических фигур.

**Задание.**

Вариант 14.

Реализовать систему классов для фигур:

1. Круг
2. Трапеция
3. Квадрат

**Пояснение задачи.**

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:

1. условие задания;
2. UML диаграмму разработанных классов;
3. текстовое обоснование проектных решений;
4. реализацию классов на языке С++.

**Текстовое обоснование разработанных классов.**

class Point – класс, который хранит координаты x,y точки, также содержит методы с помощью которых можно установить либо получить координату.

class Colour – класс, содержащий значения RGB, которые характеризуют цвет фигуры, также содержит методы с помощью которых можно получить информацию о цвете.

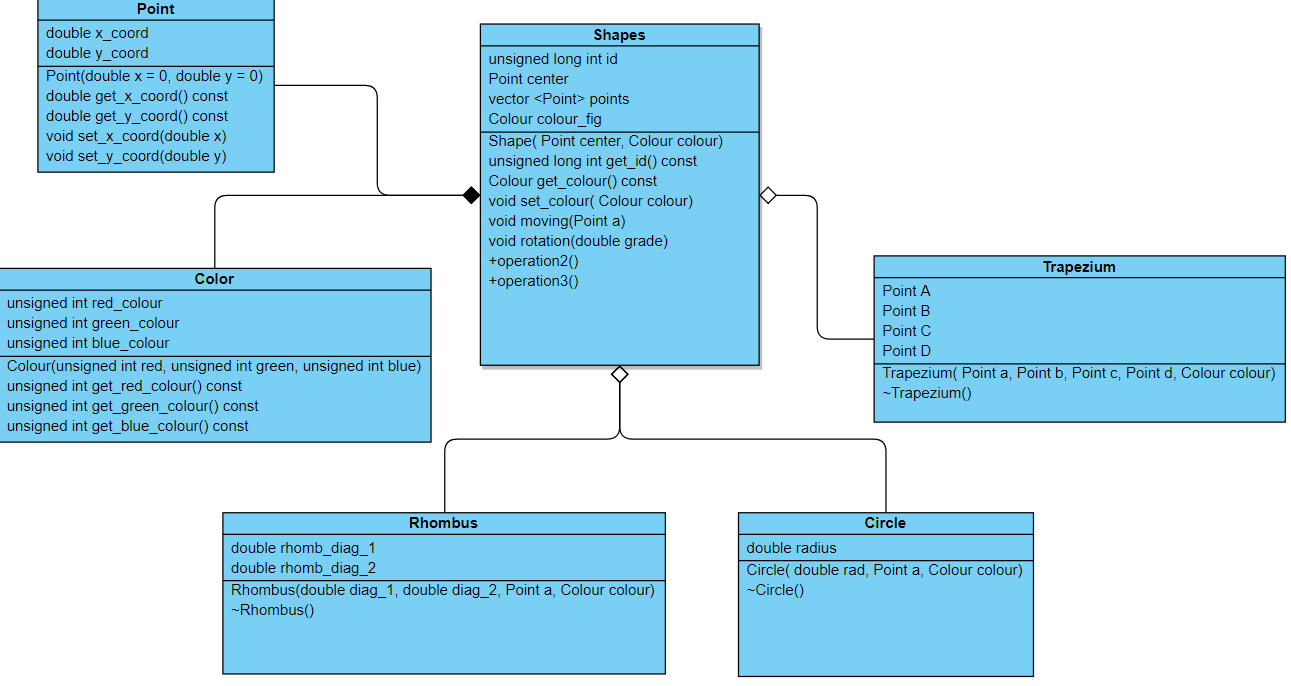
class Shape – абстрактный класс, содержащий методы для перемещения в указанные координаты, поворота на заданный угол, масштабирования на заданный коэффициент, установки и получения цвета, а также оператор вывода в поток. Также содержит информацию о фигуре: номер, координата центра, вектор с вершинами фигуры и цвет.

class Rhombus – класс, наследуемый от абстрактного Shape. Содержит 2 дополнительных защищённых поля для хранения длин диагоналей. Также содержит переопределённые методы масштабирования и метод вывода информации о фигуре.

class Circle – класс, наследуемый от абстрактного Shape. Содержит 1 защищённое поле для хранения радиуса круга. Также содержит переопределённые методы, как в классе вышеописанном классе.

class Trapezium – класс, наследуемый от абстрактного Shape. Содержит 4 дополнительных защищённых поля для хранения вершин трапеции. Также содержит переопределённые методы, как в классе вышеописанном классе.

**Диаграмма.**

****

**Выводы.**

В результате выполнения лабораторной работы ознакомился с понятиями наследование, полиморфизм, абстрактный класс, изучил виртуальные функции, принцип их работы, способ организации в памяти, раннее и позднее связывания в языке C++. Также выполнил индивидуальное задание для данных к л/р геометрических фигур.

Приложение А

Код программы

**Lab1.cpp**

[#include](https://vk.com/im?sel=159964878&st=%23include) <iostream>  
[#include](https://vk.com/im?sel=159964878&st=%23include) <vector>  
[#include](https://vk.com/im?sel=159964878&st=%23include) <cmath>  
  
[#define](https://vk.com/im?sel=159964878&st=%23define) PI 3.14159265359  
  
using namespace std;  
  
class Point  
{  
  
double x\_coord;  
double y\_coord;  
  
public:  
  
Point(double x = 0, double y = 0) : x\_coord(x), y\_coord(y) {};  
  
double get\_x\_coord() const  
{  
return x\_coord;  
}  
  
double get\_y\_coord() const  
{  
return y\_coord;  
}  
  
void set\_x\_coord(double x)  
{  
this->x\_coord = x;  
}  
  
void set\_y\_coord(double y)  
{  
this->y\_coord = y;  
}  
  
};  
  
class Colour  
{  
  
unsigned int red\_colour;  
unsigned int green\_colour;  
unsigned int blue\_colour;  
  
public:  
  
Colour(unsigned int red, unsigned int green, unsigned int blue) : red\_colour(red), green\_colour(green), blue\_colour(blue) {};  
  
unsigned int get\_red\_colour() const  
{  
return red\_colour;  
}  
  
unsigned int get\_green\_colour() const  
{  
return green\_colour;  
}  
  
unsigned int get\_blue\_colour() const  
{  
return blue\_colour;  
}  
  
};  
  
class Shape  
{  
  
protected:  
  
unsigned long int id;  
Point center;  
vector <Point> points;  
Colour colour\_fig;  
  
public:  
  
Shape( Point center, Colour colour) : center(center), colour\_fig(colour)  
{  
static long int i = 0;  
id = i;  
i++;  
}  
  
unsigned long int get\_id() const  
{  
return id;  
}  
  
Colour get\_colour() const  
{  
return colour\_fig;  
}  
  
void set\_colour( Colour colour)  
{  
this->colour\_fig = colour;  
}  
  
void moving(Point a)  
{  
double offset\_x = a.get\_x\_coord() - center.get\_x\_coord();  
double offset\_y = a.get\_y\_coord() - center.get\_y\_coord();  
for(size\_t i = 0; i < points.size(); i++)  
{  
double new\_x = points[i].get\_x\_coord() + offset\_x;  
double new\_y = points[i].get\_y\_coord() + offset\_y;  
points[i].set\_x\_coord(new\_x);  
points[i].set\_y\_coord(new\_y);  
}  
center = a;  
}  
  
void rotation(double grade)  
{  
double grade\_in\_rad = grade\*PI/180.0;  
for (size\_t i = 0; i < points.size(); i++) {  
double x = center.get\_x\_coord()+(points[i].get\_x\_coord()-center.get\_x\_coord())\*cos(grade\_in\_rad)-(points[i].get\_y\_coord()-center.get\_y\_coord())\*sin(grade\_in\_rad);  
double y = center.get\_y\_coord() + (points[i].get\_x\_coord() - center.get\_x\_coord())\*sin(grade\_in\_rad) + (points[i].get\_y\_coord() - center.get\_y\_coord())\*cos(grade\_in\_rad); ;  
points[i].set\_x\_coord(x);  
points[i].set\_y\_coord(y);  
}  
}  
  
virtual void scaling(double coefficient)=0;  
  
virtual ostream& figure\_info(ostream& stream, Shape& figure) = 0;  
friend ostream& operator « (ostream& stream, Shape& figure) {  
return figure.figure\_info(stream, figure);  
}  
  
};  
  
class Rhombus : public Shape  
{  
  
double rhomb\_diag\_1;  
double rhomb\_diag\_2;  
  
public:  
  
Rhombus(double diag\_1, double diag\_2, Point a, Colour colour) : Shape(center, colour) , rhomb\_diag\_1(diag\_1) , rhomb\_diag\_2(diag\_2)  
{  
points.push\_back(a);  
points.push\_back(Point(a.get\_x\_coord() - (rhomb\_diag\_2 / 2), a.get\_y\_coord() + (rhomb\_diag\_1 / 2)));  
points.push\_back(Point(a.get\_x\_coord(), a.get\_y\_coord() + rhomb\_diag\_1));  
points.push\_back(Point(a.get\_x\_coord() + (rhomb\_diag\_2 / 2), a.get\_y\_coord() + (rhomb\_diag\_1 / 2)));  
center.set\_x\_coord((points[1].get\_x\_coord() + points[3].get\_x\_coord()) / 2);  
center.set\_y\_coord((points[0].get\_y\_coord() + points[2].get\_y\_coord()) / 2);  
}  
  
~Rhombus()  
{  
points.clear();  
}  
  
void scaling(double coeff) override  
{  
double x\_coord, y\_coord;  
for(size\_t i = 0; i < points.size(); i++)  
{  
x\_coord = center.get\_x\_coord() + (points[i].get\_x\_coord() - center.get\_x\_coord())[\*coeff](https://vk.com/coeff);  
y\_coord = center.get\_y\_coord() + (points[i].get\_y\_coord() - center.get\_y\_coord())[\*coeff](https://vk.com/coeff);  
points[i].set\_x\_coord(x\_coord);  
points[i].set\_y\_coord(y\_coord);  
}  
}  
  
ostream& figure\_info(ostream& stream, Shape& figure) override  
{  
stream « "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" « endl;  
stream « "Информация о фигуре:" « endl;  
stream « "Фигура — ромб" « endl;  
stream « "id: " « figure.get\_id() « endl;  
stream « "Цвет фигуры: " « figure.get\_colour().get\_red\_colour() « " " « figure.get\_colour().get\_green\_colour() « " " « figure.get\_colour().get\_blue\_colour() « endl;  
stream « "Центр фигуры находится в точке с

координатами: " « "(" « center.get\_x\_coord() « ";" « center.get\_y\_coord() « ")" « endl;  
stream « "Фигура находится в следующих координатах: " « endl;  
for(size\_t i = 0; i < points.size(); i++)  
{  
stream « "(" « points[i].get\_x\_coord() « ";" « points[i].get\_y\_coord() « ")" « endl;  
}  
stream « "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" « endl;  
return stream;  
}  
  
};  
  
class Circle : public Shape  
{  
  
double radius;  
  
public:  
  
Circle( double rad, Point a, Colour colour) : Shape(center, colour), radius(rad)  
{  
center = a;  
points.push\_back(Point(a.get\_x\_coord() - radius, a.get\_y\_coord()));  
points.push\_back(Point(a.get\_x\_coord(), a.get\_y\_coord() + radius));  
points.push\_back(Point(a.get\_x\_coord() + radius, a.get\_y\_coord()));  
points.push\_back(Point(a.get\_x\_coord(), a.get\_y\_coord() - radius));  
  
}  
  
void scaling(double coeff) override  
{  
double x\_coord, y\_coord;  
for(size\_t i = 0; i < points.size(); i++)  
{  
x\_coord = center.get\_x\_coord() + (points[i].get\_x\_coord() - center.get\_x\_coord())[\*coeff](https://vk.com/coeff);  
y\_coord = center.get\_y\_coord() + (points[i].get\_y\_coord() - center.get\_y\_coord())[\*coeff](https://vk.com/coeff);  
points[i].set\_x\_coord(x\_coord);  
points[i].set\_y\_coord(y\_coord);  
}  
}  
  
~Circle()  
{  
points.clear();  
}  
  
ostream& figure\_info(ostream& stream, Shape& figure) override  
{  
stream « "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" « endl;  
stream « "Информация о фигуре:" « endl;  
stream « "Фигура — круг" « endl;  
stream « "id: " « figure.get\_id() « endl;  
stream « "Цвет фигуры: " « figure.get\_colour().get\_red\_colour() « " " « figure.get\_colour().get\_green\_colour() « " " « figure.get\_colour().get\_blue\_colour() « endl;  
stream « "Центр фигуры находится в точке с координатами: " « "(" « center.get\_x\_coord() « ";" « center.get\_y\_coord() « ")" « endl;  
stream « "Фигура находится в следующих координатах: " « endl;  
for(size\_t i = 0; i < points.size(); i++)  
{  
stream « "(" « points[i].get\_x\_coord() « ";" « points[i].get\_y\_coord() « ")" « endl;  
}  
stream « "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" « endl;  
return stream;  
}  
  
};  
  
class Trapezium : public Shape  
{  
Point A;  
Point B;  
Point C;  
Point D;  
  
// B C  
//  
//  
// A D  
  
public:  
Trapezium( Point a, Point b, Point c, Point d, Colour colour) : Shape(center, colour) , A(a), B(b), C(c), D(d)  
{  
points.push\_back(A);  
points.push\_back(B);  
points.push\_back(C);  
points.push\_back(D);  
double k = sqrt((B.get\_x\_coord() - A.get\_x\_coord()) \* (B.get\_x\_coord() - A.get\_x\_coord()) + (B.get\_y\_coord() - A.get\_y\_coord()) \* (B.get\_y\_coord() - A.get\_y\_coord()));  
double p = sqrt((C.get\_x\_coord() - D.get\_x\_coord()) \* (C.get\_x\_coord() - D.get\_x\_coord()) + (C.get\_y\_coord() - D.get\_y\_coord()) \* (C.get\_y\_coord() - D.get\_y\_coord()));  
center.set\_x\_coord((D.get\_x\_coord() - A.get\_x\_coord()) / 2 + ((2 \* (C.get\_x\_coord() - B.get\_x\_coord()) + (D.get\_x\_coord() - A.get\_x\_coord())) \* (pow(k, 2) - pow(p, 2))) / (6 \* (pow(D.get\_x\_coord() - A.get\_x\_coord(), 2) - pow(C.get\_x\_coord() - B.get\_x\_coord(), 2))) + A.get\_x\_coord());  
center.set\_y\_coord((sqrt(pow(B.get\_x\_coord() - B.get\_x\_coord(), 2) + pow(B.get\_y\_coord() - A.get\_y\_coord(), 2))) \* ((D.get\_x\_coord() - A.get\_x\_coord()) + 2 \* (C.get\_x\_coord() - B.get\_x\_coord())) / (3 \* ((C.get\_x\_coord() - B.get\_x\_coord()) + (D.get\_x\_coord() - A.get\_x\_coord()))) + A.get\_y\_coord());  
}  
  
void scaling(double coeff) override  
{  
double x\_coord, y\_coord;  
for(size\_t i = 0; i < points.size(); i++)  
{  
x\_coord = center.get\_x\_coord() + (points[i].get\_x\_coord() - center.get\_x\_coord())[\*coeff](https://vk.com/coeff);  
y\_coord = center.get\_y\_coord() + (points[i].get\_y\_coord() - center.get\_y\_coord())[\*coeff](https://vk.com/coeff);  
points[i].set\_x\_coord(x\_coord);  
points[i].set\_y\_coord(y\_coord);  
}  
}  
  
~Trapezium()  
{  
points.clear();  
}  
  
ostream& figure\_info(ostream& stream, Shape& figure) override  
{  
stream « "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" « endl;  
stream « "Информация о фигуре:" «

endl;  
stream « "Фигура — трапеция" « endl;  
stream « "id: " « figure.get\_id() « endl;  
stream « "Цвет фигуры: " « figure.get\_colour().get\_red\_colour() « " " « figure.get\_colour().get\_green\_colour() « " " « figure.get\_colour().get\_blue\_colour() « endl;  
stream « "Центр фигуры находится в точке с координатами: " « "(" « center.get\_x\_coord() « ";" « center.get\_y\_coord() « ")" « endl;  
stream « "Фигура находится в следующих координатах: " « endl;  
for(size\_t i = 0; i < points.size(); i++)  
{  
stream « "(" « points[i].get\_x\_coord() « ";" « points[i].get\_y\_coord() « ")" « endl;  
}  
stream « "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" « endl;  
return stream;  
}  
  
};  
  
int main()  
{  
Rhombus r\_figure(10, 20 , {30, 30}, { 187, 255, 211 });  
cout « r\_figure;  
cout « "Поворот" « endl;  
r\_figure.rotation(30);  
cout « r\_figure;  
cout « "Масштабирование" « endl;  
r\_figure.scaling(2);  
cout « r\_figure;  
cout « "Перемещение" « endl;  
r\_figure.moving({ 3,6 });  
cout « r\_figure;  
Circle c\_figure( 5, {0, 0}, { 255, 255, 255 });  
cout « c\_figure;  
cout « "Масштабирование" « endl;  
c\_figure.scaling(3);  
cout « c\_figure;  
cout « "Перемещение" « endl;  
c\_figure.moving({ 1,8 });  
cout « c\_figure;  
Trapezium t\_figure({0, 0}, {3, 5}, {7, 5}, {10, 0}, { 14, 218, 111 });  
cout « t\_figure;  
cout « "Поворот" « endl;  
t\_figure.rotation(40);  
cout « t\_figure;  
cout « "Масштабирование" « endl;  
t\_figure.scaling(4);  
cout « t\_figure;  
cout « "Перемещение" « endl;  
t\_figure.moving({ 2,10 });  
cout « t\_figure;  
return 0;  
}