**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практической работе №2**

**по дисциплине «Объектно-ориентированное программирование»**

**Тема**: **Наследование**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 7382 |  | Еременко А.А. |
| 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.

**Постановка 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:

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

Вариант №10. Фигуры : прямоугольник, трапеция и равнобедренная трапеция.

**Текстовое обоснование разработанных классов.**

struct Point – структура для хранения значений точки.

struct RGB – структура для хранения значения цвета фигуры.

class Shape – базовый класс. Он содержит основные методы для геометрических фигур:

* void MoveFigure(Point new\_center)- перемещение фигуры в указанные кородинаты.
* void RotationAngle(int new\_angle) – поворот фигуры на заданный угол.
* void Coloring(int new\_R, int new\_G, int new\_B) – раскраска.
* void ShapeInfo(ostream& out) – вывод в поток.

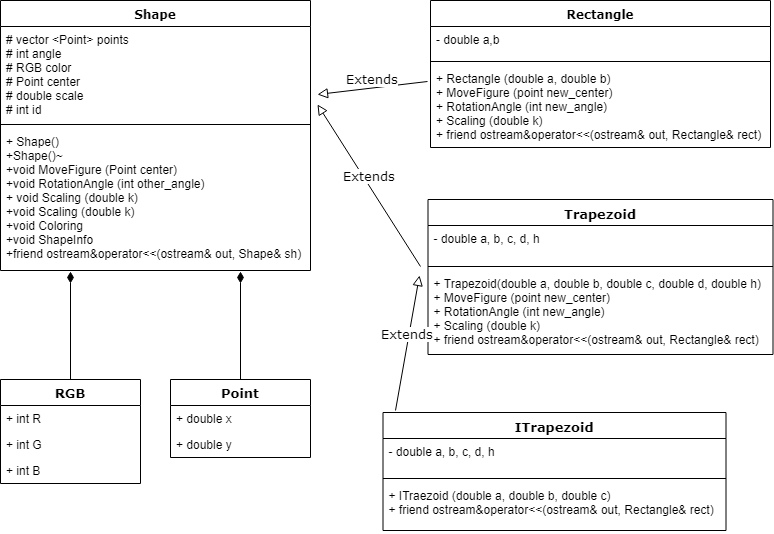
Классы Rectangle и Trapezoid – класс наследуемые от класса Shape, а класс ITrapezoid наследуется от класса Trapezoid.

**Выводы.**

В ходе выполнения лабораторной работы была спроектирована система классов для работы с геометрическими фигурами в соответствии с индивидуальным заданием. В иерархии наследования были использованы виртуальные функции, базовый класс при этом является виртуальным (класс называется виртуальным, если содержит хотя бы одну виртуальную функцию). Были реализованы методы перемещения фигуры в заданные координаты, поворота на заданный угол, масштабирования на заданный коэффициент, была реализована однозначная идентификация объекта.

**ПРИЛОЖЕНИЕ А**

UML диаграмма разработанных классов



**ПРИЛОЖЕНИЕ Б**

Исходный код программы

#include <cmath>

#include <vector>

#include <algorithm>

#include <iostream>

#include <exception>

using namespace std;

const double PI = 3.14159265358979323846264338327950288419717;

struct Point{

double x;

double y;

};

struct RGB{

int R;

int G;

int B;

};

class Shape {

protected:

static int i;

vector <Point> points;

int angle;

RGB color;

Point center;

double scale;

public:

Shape(){

center={0, 0};

color = {0, 0, 0};

scale = 1;

angle = 0;

i++;

}

// Перемещение фигуры

void MoveFigure(Point new\_center){

double dx = new\_center.x - center.x;

double dy = new\_center.y - center.y;

for(int i = 0; i < points.size(); i++){

points[i].x = points[i].x+dx;

points[i].y = points[i].y+dy;

}

center = new\_center;

}

// Поворот на угол

void RotationAngle(int new\_angle) {

angle = new\_angle;

new\_angle%=360;

double a = new\_angle\*PI/180;

double \_x, \_y;

for (unsigned int i = 0; i < points.size(); ++i) {

\_x = points[i].x;

\_y = points[i].y;

points[i].x = ( \_x - center.x ) \* cos(a) - ( \_y - center.y ) \* sin(a) + center.x;

points[i].y = ( \_x - center.x ) \* sin(a) + ( \_y - center.y ) \* cos(a) + center.y;

if(points[i].x<0.000001)

points[i].x = 0;

if(points[i].y<0.000001)

points[i].y = 0;

}

}

virtual void Scaling(double k) = 0;

// Раскраска

void Coloring(int new\_R, int new\_G, int new\_B) {

color.R = new\_R;

color.G = new\_G;

color.B = new\_B;

}

// Вывод информации

void ShapeInfo(ostream& out) {

out.precision(2);

out << "ID = "<<i<<endl;

out << "Figure centre: (" << center.x << ", " << center.y << ")\n";

out << "Angle of rotation = " << angle << endl;

out << "Color (RGB): (" << color.R << ", " << color.G << ", " << color.B << ")\n";

out << "Scale factor = " << scale << endl;

out << endl;

}

friend ostream& operator<<(ostream& out, Shape& sh){

sh.ShapeInfo(out);

return out;

}

virtual ~Shape(){}

};

// Клссс прямоугольника

class Rectangle : public Shape {

private:

double a, b;

public:

Rectangle(double a, double b)

: a(a), b(b) {

makecoord();

}

void makecoord(){

points.push\_back({-a/2, b/2});

points.push\_back({a/2, b/2});

points.push\_back({a/2, -b/2});

points.push\_back({-a/2, -b/2});

}

void Scaling(double k) {

a\*=k;

b\*=k;

scale\*=k;

points.clear();

makecoord();

}

~Rectangle(){}

};

// Класс трапеции

class Trapezoid : public Shape{

protected:

double a, b, c, d, h;

public:

Trapezoid(double a, double b, double c, double d, double h): a(a), b(b), c(c), d(d), h(h){

makecoord();

}

void makecoord(){

center.x = b/2+(2\*a+b)\*(c\*c-d\*d)/(6\*(b\*b-a\*a));

center.y = h\*(b+2\*a)/(3\*(a+b));

points.clear();

points.push\_back({0, 0});

points.push\_back({sqrt(c\*c-h\*h), h});

points.push\_back({b-sqrt(d\*d-h\*h), h});

points.push\_back({b, 0});

MoveFigure(center);

}

void Scaling(double k){

c\*=k;

h\*=k;

d\*=k;

b\*=k;

a\*=k;

makecoord();

}

~Trapezoid() {}

};

// Класс равнобедренной трапеции

class ITrapezoid:public Trapezoid{

public:

ITrapezoid(double a, double b, double c):Trapezoid(a, b, c, c,sqrt(c\*c- (b-a)/2\*(b-a)/2)){};

~ITrapezoid(){}

};

int Shape::i = 0;

int main(){

Shape \*sh;

//Тестирование

Rectangle p(4, 3);

sh = &p;

cout << p;

sh->Coloring(1, 1, 1);

sh->RotationAngle(90);

cout << p;

sh->Scaling(2);

cout << p;

Trapezoid t(3, 9, 5, 5, 4);

sh=&t;

sh->Coloring(1,1,1);

// sh->MoveFigure({1,1});

cout<<t;

// sh->Scaling(2);

cout<<t;

sh->RotationAngle(180);

cout<<t;

ITrapezoid q(3, 9, 5);

sh=&q;

cout<<q;

return 0;

}