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. 6304 _____ Иванов В.С.

Преподаватель _____ Терентьев А.О.

Санкт-Петербург
2018

Цель работы.

Изучить наследование объектов в C++.

Основные теоретические положения.

Наследование — свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом.

Каждый производный класс полностью сохраняет интерфейс родительского класса. Обратное, очевидно, неверно.

Перегрузка методов – возможность использования методов с одним именем, но с разными параметрами.

Переопределение методов – возможность языка программирования, позволяющая производному классу обеспечивать специфическую реализацию методов, уже определенных в базовом классе.

В C++ метод, обозначаемый ключевым словом “virtual”, может быть переопределен производным классом. Компилятор всегда сначала ищет возможность запустить метод, непосредственно объявленный в классе, и лишь при отсутствии такового будет искать метод в базовых классах.

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 C++.

Для разработки диаграммы классов UML необходимо использовать какой-либо онлайн редактор, например, https://yuml.me/

Ход работы.

1. Написана структура Point, отвечающий за создание точек по координатам. Включает в себя данные (переменные x и y) и конструктор.
2. Написана структура Colour, отвечающий за цвет. Включает в себя данные (переменные r, g и b) и конструктор.
3. Написан класс «Shape», описывающий абстрактную фигуру. Класс хранит следующие переменные:
    - Центральную точку типа Point
    - Угол поворота фигуры типа int
    - Цвет типа Colour
    - Набор точек типа vector<Point>

    В классе реализованы следующие методы:
    - Конструктор
    - Метод перемещения центра фигуры по координатам *void changePos (double x, double y)*
    - Метод для изменения размера фигуры *virtual void changeSize(double k)*
    - Вспомогательный метод для изменения размера *void forSize(double& k)*
    - Метод для поворота на заданный угол, вокруг центральной точки *void changeAngle(int deg)*
    - Метод изменения цвета фигуры *void changeColour(short r, short g, short b)*
    - Перегружен оператор вывода в поток *friend std::ostream& operator<<(std::ostream& out, Shape& sh)*
    - Метод для вывода информации *virtual void print(std::ostream& out)*

- Вспомогательный метод для вывода *void forPrint(std::ostream& out)*

4. Написан класс Pentagon, наследуемый от Shape.

   Хранит переменные:

   - Периметр пятиугольника типа double

   Методы класса:

   - Переопределен конструктор
   - Переопределен метод вывода
   - Переопределен метод масштабирования

5. Написан класс Rectangle, наследуемый от Shape.

   Хранит переменные:

   - Первая сторона прямоугольник типа double
   - Вторая сторона прямоугольника типа double

   Методы класса:

   - Переопределен конструктор
   - Переопределен метод вывода
   - Переопределен метод масштабирования

6. Написан класс Ellipse, наследуемый от Shape.

   Хранит переменные:

   - Большой радиус эллипса
   - Малый радиус эллипса
   - Фокусное расстояние эллипса

   Методы класса:

   - Переопределен конструктор
   - Переопределен метод вывода
   - Переопределен метод масштабирования

7. Написаны тесты для классов Shape, Pentagon, Rectangle и Ellipse.
8. Код представлен в приложении А.
9. Нарисована UML-диаграмма классов. Рисунок в приложении Б.

Выводы.

Изучено наследование классов в C++. Написан базовый класс Shape, от которого наследуются Pentagon, Rectangle и Ellipse.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ LR2.CPP

```
#include <iostream>
#include <cmath>
#include <vector>

struct Point{
    double x;
    double y;
    Point(double x, double y): x(x),y(y){}
};
struct Colour{
    short r;
    short g;
    short b;
    Colour (short r, short g, short b):r(r),g(g),b(b){}
};

class Shape{
public:
    Shape(double x, double y):ang(0), cent(x,y), col(0,0,0){}

    void changePos(double x, double y){
        for(auto& it: pts){
            it.x+=x - cent.x;
            it.y+=y - cent.y;
        }
        cent.x=x;
        cent.y=y;
    }

    void forSize(double& k){
        if(k<0){
            k=fabs(k);
        }
        cent.x*=k;
        cent.y*=k;
        for(auto& it: pts){
            it.x*=k;
            it.y*=k;
        }
    }

    virtual void changeSize(double k){};

    void changeAngle(int deg){
        ang+=deg;
        ang%=360;
        double rad=ang*M_PI/180;
        for(auto& it: pts){
            double tmpx=it.x*cos(rad)-it.y*sin(rad);
            double tmpy=it.y*cos(rad)+it.x*sin(rad);
            it.x=tmpx;
            it.y=tmpy;
        }
    }

    void changeColour(short r, short g, short b){
        col={r,g,b};
    }
```

```
    void forPrint(std::ostream& out){
        out<<"Центр. коорд. "<<cent.x<<" "<<cent.y<<std::endl;
        out<<"Угол поворота "<<ang<<std::endl;
        out<<"Точки "<<std::endl;
        int count=0;
        for(const auto& it: pts){
            count++;
            out<<count<<") ("<<it.x<<", "<<it.y<<")\n";
        }
        out<<"Цвет "<<col.r<<" "<<col.g<<" "<<col.b<<std::endl;
    }

    virtual void print(std::ostream& out){}

    friend std::ostream& operator<<(std::ostream& out, Shape& sh){
        sh.forPrint(out);
        sh.print(out);
        return out;
    }

protected:
    Point cent;
    int ang;
    Colour col;
    std:: vector<Point> pts;
};

class Pentagon : public Shape{
public:
    Pentagon(double x1, double y1, double x2, double y2, double x3, double y3, double
x4, double y4, double x5, double y5):
    Shape((x1+x2+x3+x4+x5)/5,(y1+y2+y3+y4+y5)/5), perimeter(0)
    {
        pts.push_back({x1,y1});
        pts.push_back({x2,y2});
        pts.push_back({x3,y3});
        pts.push_back({x4,y4});
        pts.push_back({x5,y5});
        for(int i=0; i<5;i++){
            perimeter+=sqrt(pow((pts[(i+1)%5].x - pts[i].x), 2) + pow((pts[(i+1)%5].y -
pts[i].y), 2));
        }
    }

    void changeSize(double k){
        forSize(k);
        perimeter*=k;
    }

    void print(std::ostream& out){
        out<<"Периметр "<< perimeter<<"\n";
    }

protected:
    double perimeter;
};

class Rectangle : public Shape{
public:
    Rectangle(double x, double y, double a, double b) : Shape(x,y), a(a), b(b){
```

```
        pts.push_back({cent.x+a/2,cent.y+b/2});
        pts.push_back({cent.x+a/2,cent.y-b/2});
        pts.push_back({cent.x-a/2,cent.y-b/2});
        pts.push_back({cent.x-a/2,cent.y+b/2});
    }

    void changeSize(double k){
        forSize(k);
        a*=k;
        b*=k;
    }

    void print(std::ostream& out){
        out<<"Первая сторона "<< a<<"\n";
        out<<"Вторая сторона "<< b<<"\n";
    }

protected:
    double a;
    double b;
};
class Ellipse : public Shape{
public:
    Ellipse(double x, double y, double R, double r) : Shape(x,y), R(R), r(r){
        pts.push_back({x+R,y});
        pts.push_back({x,y-r});
        pts.push_back({x-R,y});
        pts.push_back({x,y+r});
        focdist = std::max(R,r)*sqrt(1-
(std::min(R,r)*std::min(R,r))/(std::max(R,r)*std::max(R,r)));
    }

    void changeSize(double k){
        forSize(k);
        r*=k;
        R*=k;
        focdist*=k;
    }

    void print(std::ostream& out){
        out<<"Большой радиус "<< R<<"\n";
        out<<"Малый радиус "<<r<<"\n";
        out<<"Фокусное расстояние "<< focdist<<"\n";
    }

 protected:
    double R, r;
    double focdist;
};

int main(){
    std::cout<<"-----------\n";
    std::cout<<"Shape\n";
    std::cout<<"-----------\n";
    Shape sh(3,4);
    sh.changeAngle(380);
    sh.changeColour(5,77,133);
  //  sh.changePos(4,5);
    sh.changeSize(3);
    std::cout<<sh;
```

```
    std::cout<<"-----------\n";
    std::cout<<"Ellipse\n";
    std::cout<<"-----------\n";
    Ellipse e(0,0,2,1);
    std::cout<<e;

    std::cout<<"-----------\n";
    std::cout<<"Edited Ellipse\n";
    std::cout<<"-----------\n";
    e.changeAngle(90);
    e.changeColour(5,77,133);
    e.changePos(4,5);
    e.changeSize(3);
    std::cout<<e;


    std::cout<<"-----------\n";
    std::cout<<"Pentagon\n";
    std::cout<<"-----------\n";
    Pentagon p(0,0,1,1,1,2,-1,2,-1,1);
    std::cout<<p;

    std::cout<<"-----------\n";
    std::cout<<"Edited Pentagon\n";
    std::cout<<"-----------\n";
    p.changeAngle(90);
    p.changeColour(5,77,133);
    p.changePos(4,4);
    p.changeSize(3);
    std::cout<<p;


    std::cout<<"-----------\n";
    std::cout<<"Rectangle\n";
    std::cout<<"-----------\n";
    Rectangle r(0,0,2,3);
    std::cout<<r;

    std::cout<<"-----------\n";
    std::cout<<"Edited Rectangle\n";
    std::cout<<"-----------\n";
    r.changeAngle(90);
    r.changeColour(5,77,133);
    r.changePos(4,4);
    r.changeSize(-3);
    std::cout<<r;

    return 0;
}
```

ПРИЛОЖЕНИЕ Б

UML-ДИАГРАММА