МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ГРАЖДАНСКОЙ АВИАЦИИ  
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ  
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ

**Лабораторная работа №4**

**«НАСЛЕДОВАНИЕ И ПОЛИМОРФИЗМ В С++»**

По дисциплине: «Объектно-ориентированное программирование»

Вариант 9

Выполнил:

студент группы ПМ4-1

Фейзуллин К.М.

***Цель работы.***

*Целью* данной *лабораторной работы* является освоение приемов:

* наследования при создании классов;
* создания многоуровневой иерархической структуры классов;
* наследования от разных классов;
* вызова конструкторов базового класса;
* создания и использования виртуальных функций.

***Задание на выполнение лабораторной работы.***

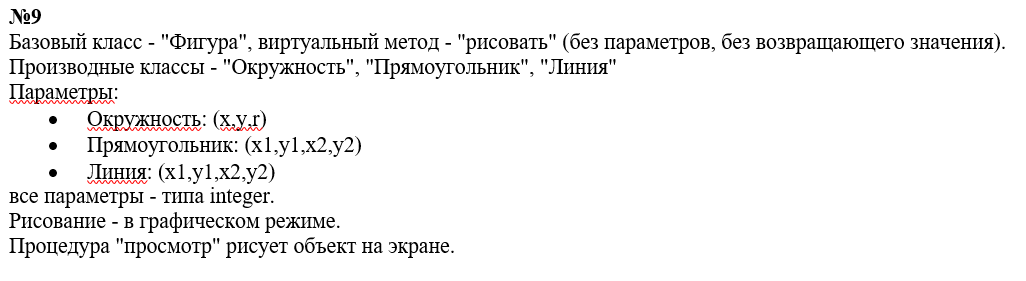
Разработать программу в соответствии со своим вариантом, включающую:

1. объявление базового и производных классов (в том числе для построения иерархической структуры, содержащей не менее двух уровней производных классов);
2. защищенные члены класса;
3. виртуальные функции (например, для процедуры печати);
4. конструктор базового класса и конструктор производного класса;
5. класс, наследующий от двух классов (допускается данный пункт выполнить в отдельной программе).

Выделение функций обработки и распределение по классам (базовый/производный) выполняется самостоятельно с учётом разумности и целесообразности.

Результаты работы должны быть выведены на экран.

Исходные данные должны быть подобраны таким образом, чтобы можно было проверить правильность работы программы.

***Вариант №9***

***Код программы***

#include <iostream>

#include "graphics.h"

using namespace std;

class Color

{

protected:

    int \_color = 15;

public:

    Color() {};

    Color(const int c) { this->\_color = c; }

    Color(const string s);

    void SetColor(const int c);

    void SetColor(const string s);

    string GetColor();

};

class TwoDFigure : public Color

{

protected:

    int\* \_cords = NULL;

    int \_cordsCount = 0;

public:

    TwoDFigure() {};

    TwoDFigure(int\* cords, int n);

    TwoDFigure(int\* cords, int n, int c);

    TwoDFigure(int\* cords, int n, const string s);

    virtual void Draw() = 0;

    virtual void SetCords(int \*cords) = 0;

    int\* GetCords();

    int GetCordsCount();

};

class TwoDLine : public TwoDFigure

{

public:

    TwoDLine(int\* cords) : TwoDFigure(cords, 2) { };

    void SetCords(int\* cords);

    void Draw();

private:

    const int \_limit = 2;

};

class TwoDRectangle : public TwoDFigure

{

public:

    TwoDRectangle(int\* cords) : TwoDFigure(cords, 2) {};

    void SetCords(int\* cords);

    void Draw();

private:

    const int \_limit = 2;

};

class TwoDCircle : public TwoDFigure

{

public:

    TwoDCircle(int\* cords, int radius) : TwoDFigure(cords, 1) { this->\_rad = radius; };

    void SetCords(int\* cords);

    void SetRadius(int r);

    int GetRadius();

    void Draw();

private:

    int \_rad = 0;

    const int \_limit = 1;

};

int main()

{

    int mas[4] = { 10, 10, 300, 300 };

    int mas2[2] = { 300, 300 };

    int radius = 50;

    TwoDLine l(mas);

    l.SetColor("red");

    l.Draw();

    TwoDRectangle rect(mas);

    rect.SetColor("blue");

    rect.Draw();

    TwoDCircle crcl(mas2, radius);

    crcl.SetColor("yellow");

    crcl.Draw();

    return 0;

}

int\* TwoDFigure::GetCords() {

    int n = this->\_cordsCount;

    int \*\_new = new int[n \* 2];

    for (int i = 0; i < n; i++)

    {

        \_new[i \* 2] = \_cords[i \* 2];

        \_new[i \* 2 + 1] = \_cords[i \* 2 + 1];

    }

    return \_new;

}

TwoDFigure::TwoDFigure(int\* cords, int n) : Color()

{

    if (cords != NULL)

    {

        this->\_cordsCount = n;

        this->\_cords = new int[n \* 2];

        for (int i = 0; i < n; i++)

        {

            this->\_cords[i \* 2] = cords[i \* 2];

            this->\_cords[i \* 2 + 1] = cords[i \* 2 + 1];

        }

    }

}

TwoDFigure::TwoDFigure(int\* cords, int n, int c) : Color(c)

{

    if (cords != NULL)

    {

        this->\_cordsCount = n;

        this->\_cords = new int[n \* 2];

        for (int i = 0; i < n; i++)

        {

            this->\_cords[i \* 2] = cords[i \* 2];

            this->\_cords[i \* 2 + 1] = cords[i \* 2 + 1];

        }

    }

}

TwoDFigure::TwoDFigure(int\* cords, int n, const string s) : Color(s)

{

    if (cords != NULL)

    {

        this->\_cordsCount = n;

        this->\_cords = new int[n \* 2];

        for (int i = 0; i < n; i++)

        {

            this->\_cords[i \* 2] = cords[i \* 2];

            this->\_cords[i \* 2 + 1] = cords[i \* 2 + 1];

        }

    }

}

int TwoDFigure::GetCordsCount()

{

    int n = \_cordsCount;

    return n;

}

void TwoDLine::SetCords(int \* cords)

{

    for (int i = 0; i < this->\_limit; i++) {

        this->\_cords[i \* 2] = cords[i \* 2];

        this->\_cords[i \* 2 + 1] = cords[i \* 2 + 1];

    }

}

void TwoDLine::Draw()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\TC\\BGI");

    setcolor(\_color);

    line(\_cords[0], \_cords[1], \_cords[2], \_cords[3]);

    system("pause");

    closegraph();

}

void TwoDRectangle::SetCords(int\* cords)

{

    for (int i = 0; i < this->\_limit; i++) {

        this->\_cords[i \* 2] = cords[i \* 2];

        this->\_cords[i \* 2 + 1] = cords[i \* 2 + 1];

    }

}

void TwoDRectangle::Draw()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\TC\\BGI");

    setcolor(\_color);

    rectangle(\_cords[0], \_cords[1], \_cords[2], \_cords[3]);

    system("pause");

    closegraph();

}

void TwoDCircle::SetCords(int\* cords)

{

    for (int i = 0; i < this->\_limit; i++) {

        this->\_cords[i \* 2] = cords[i \* 2];

        this->\_cords[i \* 2 + 1] = cords[i \* 2 + 1];

    }

}

void TwoDCircle::SetRadius(int r)

{

    this->\_rad = r;

}

int TwoDCircle::GetRadius()

{

    int r = this->\_rad;

    return r;

}

void TwoDCircle::Draw()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\TC\\BGI");

    setcolor(\_color);

    circle(this->\_cords[0], this->\_cords[1], this->\_rad);

    system("pause");

    closegraph();

}

Color::Color(const string s)

{

    if (s == "black")

    {

        this->\_color = 0;

        return;

    }

    if (s == "blue"){

        this->\_color = 1;

        return;

    }

    if (s == "green") {

        this->\_color = 2;

        return;

    }

    if (s == "cyan") {

        this->\_color = 3;

        return;

    }

    if (s == "red") {

        this->\_color = 4;

        return;

    }

    if (s == "magenta") {

        this->\_color = 5;

        return;

    }

    if (s == "brown") {

        this->\_color = 6;

        return;

    }

    if (s == "yellow") {

        this->\_color = 14;

        return;

    }

    if (s == "white") {

        this->\_color = 15;

        return;

    }

    this->\_color = 15;

}

void Color::SetColor(const int c)

{

    this->\_color = c;

}

void Color::SetColor(const string s)

{

    if (s == "black")

    {

        this->\_color = 0;

        return;

    }

    if (s == "blue") {

        this->\_color = 1;

        return;

    }

    if (s == "green") {

        this->\_color = 2;

        return;

    }

    if (s == "cyan") {

        this->\_color = 3;

        return;

    }

    if (s == "red") {

        this->\_color = 4;

        return;

    }

    if (s == "magenta") {

        this->\_color = 5;

        return;

    }

    if (s == "brown") {

        this->\_color = 6;

        return;

    }

    if (s == "yellow") {

        this->\_color = 14;

        return;

    }

    if (s == "white") {

        this->\_color = 15;

        return;

    }

}

string Color::GetColor()

{

    if (\_color == 0)

    {

        return string("black");

    }

    if (\_color == 1)

    {

        return string("blue");

    }

    if (\_color == 2)

    {

        return string("green");

    }

    if (\_color == 3)

    {

        return string("cyan");

    }

    if (\_color == 4)

    {

        return string("red");

    }

    if (\_color == 5)

    {

        return string("magenta");

    }

    if (\_color == 6)

    {

        return string("brown");

    }

    if (\_color == 14)

    {

        return string("yellow");

    }

    if (\_color == 15)

    {

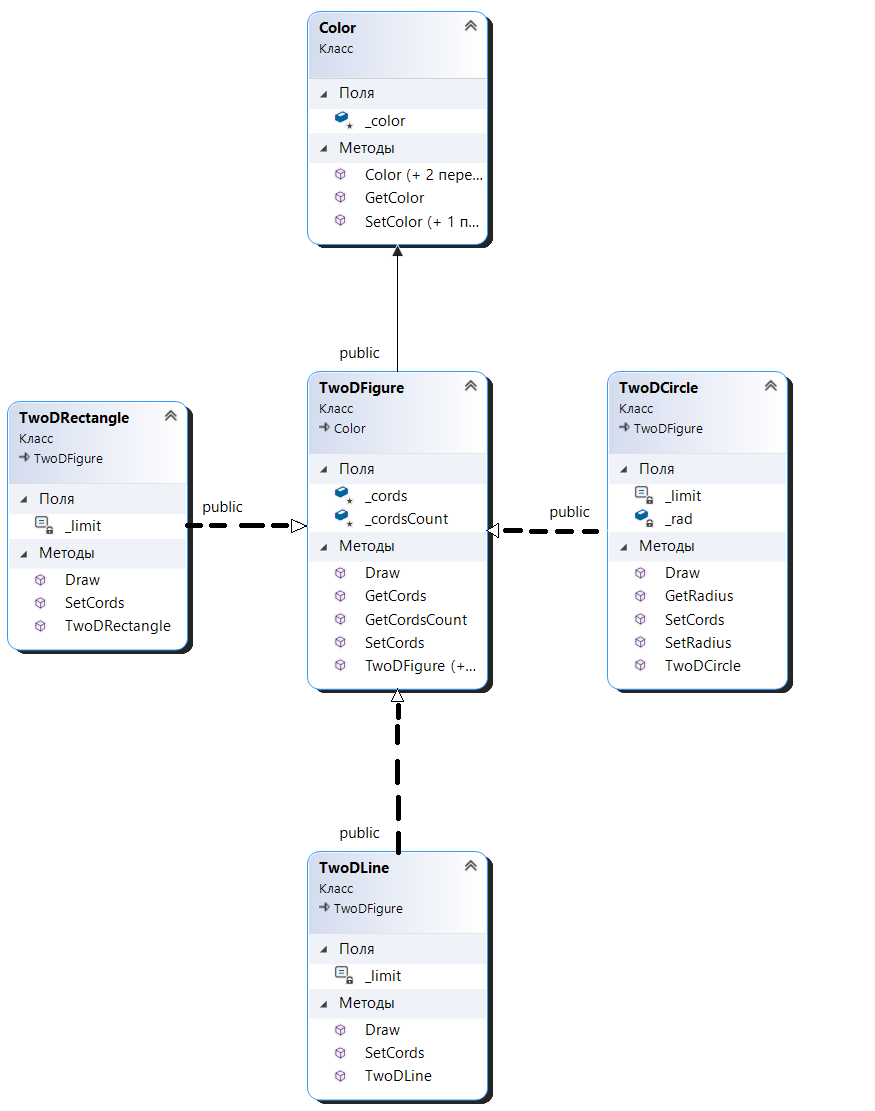
        return string("white");

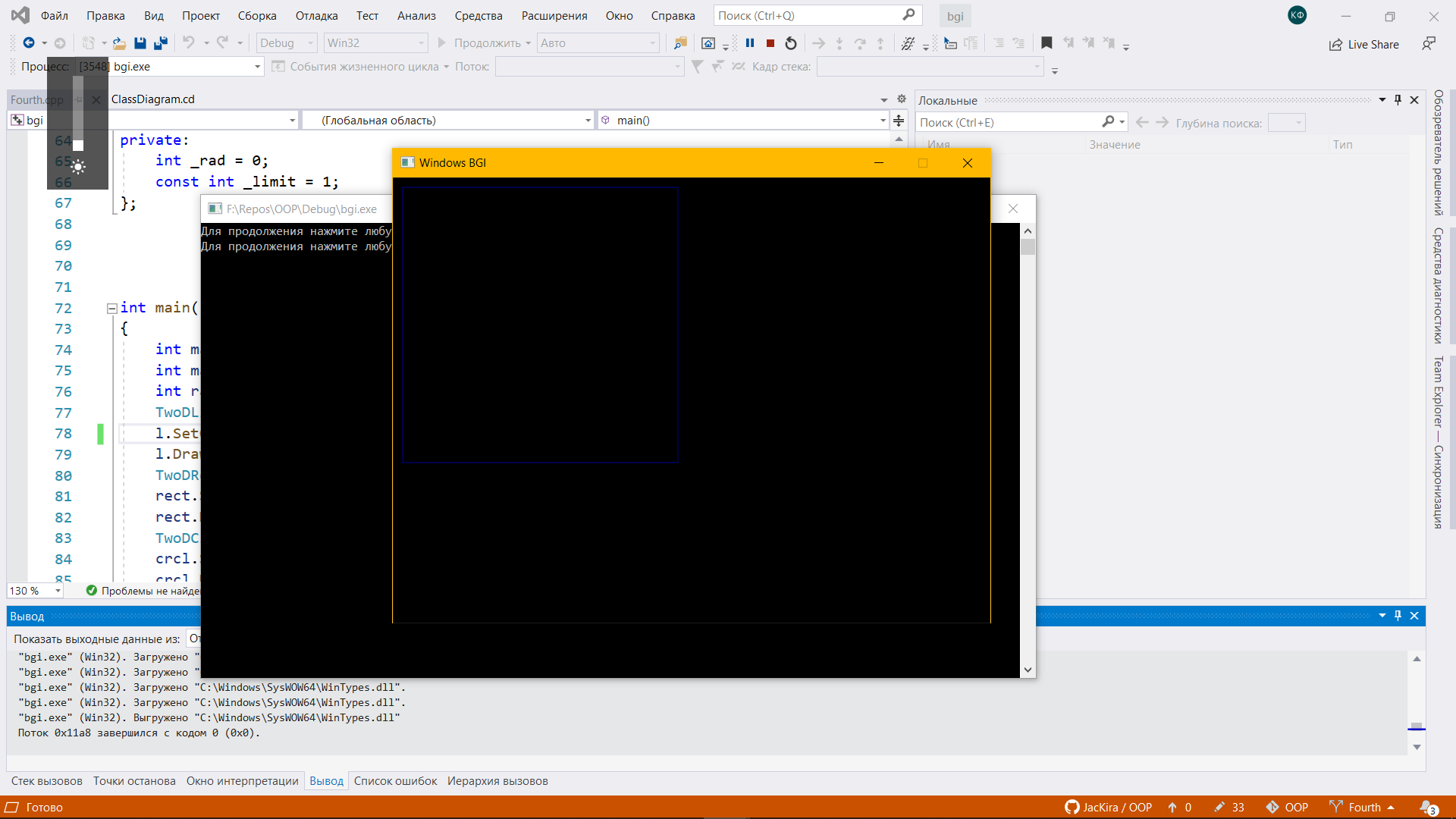
    }

    return string("dunno");

}

# ***Схема взаимодействия классов***



***Результаты работы программы***

