

Методические указания для СРС по выполнению лабораторных работ по дисциплине Объектно-ориентированное программирование

Тема: **Виртуальные методы, наследование**

I. Базовый класс для всех вариантов:

```
class Figure
{
    int c; // цвет
    bool visible;
protected:
    int x,y; // базовая точка
    virtual void draw();
public:
    Figure(int c, int x, int y);
    ~Figure();
    void move(int dx, int dy); // сместить фигуру на (dx,dy)
        // видимая фигура гасится, затем рисуется в другом месте
        // у невидимой просто меняются поля x,y
    void setcolor(int c); // установить цвет фигуры
        // видимая фигура рисуется новым цветом
        // у невидимой просто меняется поле c
    int getcolor(); // получить цвет
    void hide(); // спрятать: нарисовать черный прямоугольник
        // по размерам area()
    void show(); // показать
    bool invisible(); // видима?
    virtual void area(int &x1,int &y1,int &x2,int &y2);
        // получить размеры прямоугольной области, содержащей фигуру
};
```

Определить реализацию методов класса Figure.

Методы area и draw нужно определить как чисто виртуальные.

Как нужно определить деструктор Figure и производных классов, чтобы видимый объект исчезал с экрана при уничтожении?

Определить производный класс

Варианты заданий

Вариант 1

Ромб: Romb(цвет линий, x и y центра, длина, высота)

Вариант 2

Эллипс: Ellipse(цвет линий, x и y центра, радиус1, радиус2)

Остальные варианты <https://ipc.susu.ru/20768-4.html>

Определить дополнительный метод в производном классе для изменения размеров:

void setsizes(длина, высота);

или void setsizes(длина, высота, радиус);

или void setsizes(радиус, угол1, угол2);

и т.д., т.е. изменение значений, указываемых в аргументах конструктора, начиная с четвертого.

От написанного класса произвести новый дочерний класс - закрашенная фигура.

Например, закрашенный ромб (FillRomb ← Romb ← Figure).

Добавить к параметрам конструктора цвет заполнения.

Определить дополнительный метод для изменения цвета заполнения:

```
void setfillcolor(int c);
```

2. Реализовать main с тестами

Динамически создать две фигуры 2 разных классов, адреса объектов сохранить в переменных типа Figure *. Вызвать все методы для каждой из фигур, перед вызовом методов, определенных в производных классах, выполнить преобразование к указателю на производный класс с помощью dynamic_cast с проверкой:

```
if(Romb *r=dynamic_cast<Romb*>(o1)) r->setsize(100,50);
```

3. Написать отчет

- Постановка задачи

- Описание интерфейса классов (class {} и комментарии ко всем полям, методам и функциям)

- Описание тестов для проверки классов (main с комментариями, какие действия выполнялись, полученные результаты))

- Листинг реализации классов (реализация методов и функций)

4. Отправить отчет

Пример решения задания 3

Определите абстрактный класс Figure с методами для изменения видимости, координат и цвета линий фигуры и чисто виртуальными методами для рисования и получения координат прямоугольной области, которая содержит фигуру (для удаления фигуры с экрана рисуем черный прямоугольник по полученным координатам).

Определите производный класс Page (прямоугольник с загнутым углом) с методом изменения размеров и класс производный от него FillPage, у которого есть метод изменения цвета внутренних областей фигуры.

Напишите программу для тестирования реализованного класса.

Базовый класс Figure - одинаковый во всех заданиях

```
class Figure {
    int c; //цвет
    bool visible; // видимость
protected:
    int x, y; //базовая точка
    virtual void draw() = 0; // нарисовать
public:
    Figure(int c, int x, int y):c(c), visible(0), x(x), y(y) {}
    virtual ~Figure() {}
    void move(int x, int y); //сместить фигуру в (x, y)
    void setcolor(int c); //установить цвет фигуры, видимая рисуется, у невидимой
меняется цвет
    int getcolor() const { return c; } //получить цвет
    void hide(); //спрятать
    void show(); //показать
```

```

    bool isvisible() const { return visible; } //видима?
    virtual void area(int &x1, int &y1, int &x2, int &y2) const = 0; //размеры
//области, содержащей фигуру
};
void Figure::setcolor(int c) {
    this->c=c;
    if (visible) draw();
}
void Figure::move(int x, int y) {
    bool f=visible;
    if (f) hide();
    this->x=x;
    this->y=y;
    if (f) show();
}
void Figure::hide() {
    if (!visible) return;
    int x1, y1, x2, y2;
    area(x1, y1, x2, y2);
    setfillstyle(SOLID_FILL, BLACK);
    bar(x1, y1, x2, y2);
    visible=0;
}
void Figure::show() {
    if (visible) return;
    visible=1;
    draw();
}

```

Производные классы

```

class Page: public Figure {
protected:
    int w, h, r; //ширина, высота, размер сгиба
    void draw();
public:
    Page(int c, int x, int y, int w, int h, int r): Figure(c, x, y), w(w), h(h),
r(r) {}
    ~Page() { hide(); }
    void setsizes(int w, int h, int r); //изменение длины и высоты ромба
    void area(int &x1, int &y1, int &x2, int &y2) const; //область, где
нарисована фигура
};
void Page::setsizes(int w, int h, int r) {
    bool f=isvisible();
    if (f) hide();
    this->w=w;
    this->h=h;
    this->r=r;
    if (f) show();
}
void Page::area(int &x1, int &y1, int &x2, int &y2) const {
    x1=x;
    y1=y;
    x2=x+w;
    y2=y+h;
}
void Page::draw() {
    ::setcolor(getcolor());
    moveto(x+r, y);
    lineto(x+w,y);
    lineto(x+w, y+h);
    lineto(x, y+h);
}

```

```

        lineto(x, y+r);
        arc(x,y,r,270,360);
    }
class FillPage: public Page {
protected:
    int fc; //Цвет
    void draw();
public:
    FillPage(int c, int x, int y, int w, int h, int r, int fc):Page(c, x, y, w,
h, r), fc(fc) {}
    void setfillcolor(int); //изменить цвет
};
void FillPage::draw() {
    int p[10]={x+r, y, x+w,y, x+w,y+h, x,y+h, x,y+r };
    setfillstyle(SOLID_FILL, fc);
    fillpoly(5, p);
    Page::draw();
}
void FillPage::setfillcolor(int c) {
    fc=c;
    if (isvisible()) draw();
}
}

```

Пример проверки, задача - вызвать все методы и продемонстрировать работу виртуальных методов

```

int main() {
    initwindow(640, 480);
    Figure *a=new Page(GREEN, 110, 110, 157, 112,20);
    Figure *b=new FillPage(YELLOW, 200, 300, 100, 175, 15, BROWN);
    a->show();
    b->show();
    getch();
    a->move(90, 90);
    b->move(75, 115);
    getch();
    a->setcolor(WHITE);
    b->setcolor(RED);
    getch();
    if(Page *r=dynamic_cast<Page*>(a)) r->setsizes(100,130,15);
    if(Page *r=dynamic_cast<Page*>(b)) r->setsizes(80,150,10);
    getch();
    if(FillPage *r=dynamic_cast<FillPage*>(a)) r->setfillcolor(GREEN);
    if(FillPage *r=dynamic_cast<FillPage*>(b)) r->setfillcolor(GREEN);
    getch();
    delete a;
    delete b;
    getch();
    return 0;
}

```

Южно-Уральский государственный университет (НИУ)
Институт естественных и точных наук
Кафедра «Прикладная математика и программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
по дисциплине «Объектно-ориентированное программирование»

Автор работы
студент группы ЕТ-212
_____ А.А.Александрова
_____ 2019 г.

Работа зачтена с оценкой

_____ А.К.Демидов
_____ 2019 г.

Челябинск, 2019

1 Постановка задачи

I. Базовый класс для всех вариантов:

```
class Figure
{
    int c; // цвет
    bool visible;
protected:
    int x,y; // базовая точка
    virtual void draw();
public:
    Figure(int c, int x, int y);
    ~Figure();
    void move(int x, int y); // сместить фигуру в (x,y)
    void setcolor(int c); // установить цвет фигуры
                          // видимая фигура рисуется
                          //новым цветом
                          // у невидимой просто меняется поле c
    int getcolor(); // получить цвет
    void hide();    // спрятать: нарисовать черный
                  //прямоугольник
                  // по размерам area()
    void show();    // показать
    bool isvisible(); // видима?
    virtual void area(int &x1,int &y1,int &x2,int &y2);
                  // получить размеры прямоугольной
                  //области, содержащей фигуру
};
```

Определить реализацию методов класса Figure.

Методы area и draw нужно определить как чисто виртуальные.

Как нужно определить деструктор Figure и производных классов, чтобы видимый объект исчезал с экрана при уничтожении?

Определить производный класс

1. Ромб

Romb(цвет линий, x и y центра, длина, высота)

Определить дополнительный метод в производном классе для изменения размеров:

```
void setsizes(длина, высота);
```

или void setsizes(длина, высота, радиус);

или void setsizes(радиус, угол1, угол2);

и т.д., т.е. изменение значений, указываемых в аргументах конструктора, начиная с четвертого.

От написанного класса произвести новый дочерний класс - закрашенная фигура.

Например, закрашенный ромб (FillRomb ← Romb ← Figure).

Добавить к параметрам конструктора цвет заполнения.

Определить дополнительный метод для изменения цвета заполнения:

```
void setfillcolor(int c);
```

II. Реализовать main с тестами

Динамически создать две фигуры 2 разных классов, адреса объектов сохранить в переменных типа Figure *. Вызвать все методы для каждой из фигур, перед вызовом методов,

определенных в производных классах, выполнить преобразование к указателю на производный класс с помощью `dynamic_cast` с проверкой:

```
if (Romb *r=dynamic_cast<Romb*>(o1))  
r->setsizes(100,50);
```

2 Описание интерфейса класса

```
class Figure {
    int c; //цвет
    bool visible; // видимость
protected:
    int x, y; //базовая точка
    virtual void draw() = 0; // нарисовать
public:
    Figure(int c, int x, int y):c(c), visible(0), x(x), y(y) {} // конструктор
    virtual ~Figure() {} // деструктор
    void move(int x, int y); //сместить фигуру в x, y
    void setcolor(int c); //установить цвет фигуры
    int getcolor() const { return c; } //получить цвет
    void hide(); //спрятать
    void show(); //показать
    bool isvisible() const { return visible; } //видима?
    virtual void area(int &x1, int &y1, int &x2, int &y2) const = 0;
        // границы области, содержащей фигуру
};
class Romb: public Figure {
protected:
    int l, // длина ромба
        h; // высота ромба
    void draw(); // нарисовать
public:
    Romb(int c, int x, int y, int l, int h): Figure(c, x, y), l(l), h(h) {} // конструктор
    ~Romb() { hide(); } // деструктор
    void setsizes(int l, int h); //изменение длины и высоты ромба
    void area(int &x1, int &y1, int &x2, int &y2) const;
        // границы области, где нарисована фигура
};
class FillRomb: public Romb {
protected:
    int fc; // цвет закраски
    void draw(); // нарисовать
public:
    FillRomb(int c, int x, int y, int l, int h, int fc):Romb(c, x, y, l, h), fc(fc) {}
    void setfillcolor(int); //изменить цвет закраски
};
```

3 Описание тестов для проверки классов

```
int main() {
    initwindow(640, 480);
    Figure *a=new Romb(GREEN, 110, 110, 157, 112);
    Figure *b=new FillRomb(YELLOW, 200, 300, 100, 75, BROWN);
    a->show();
```

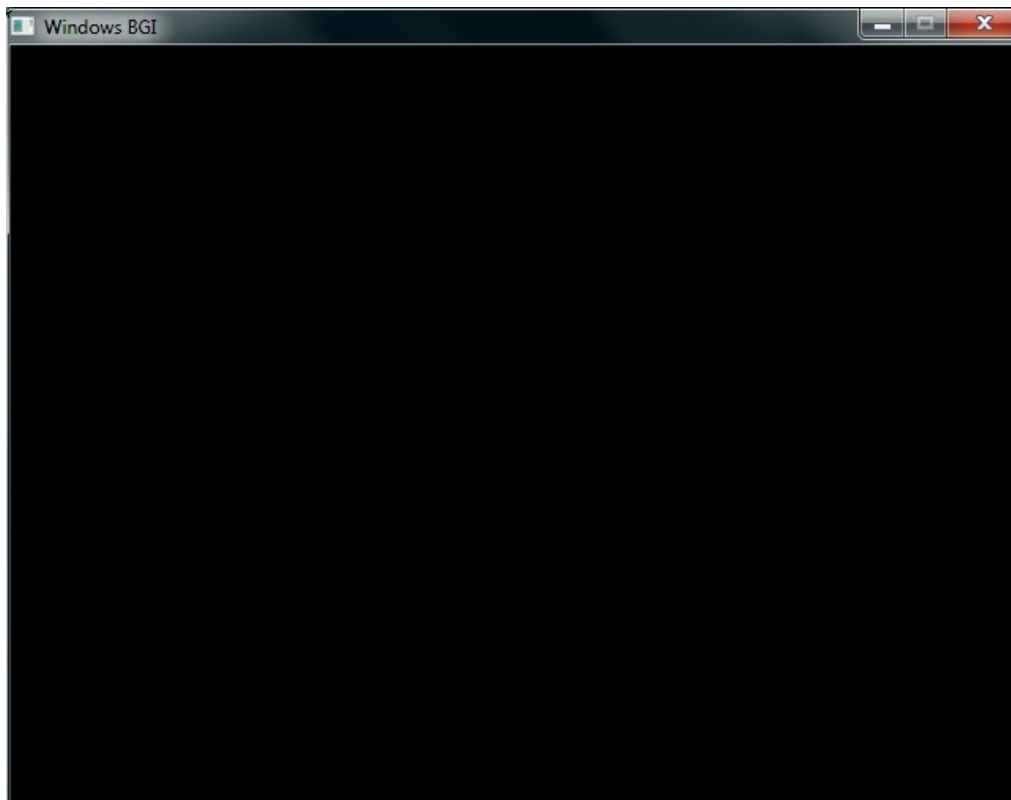
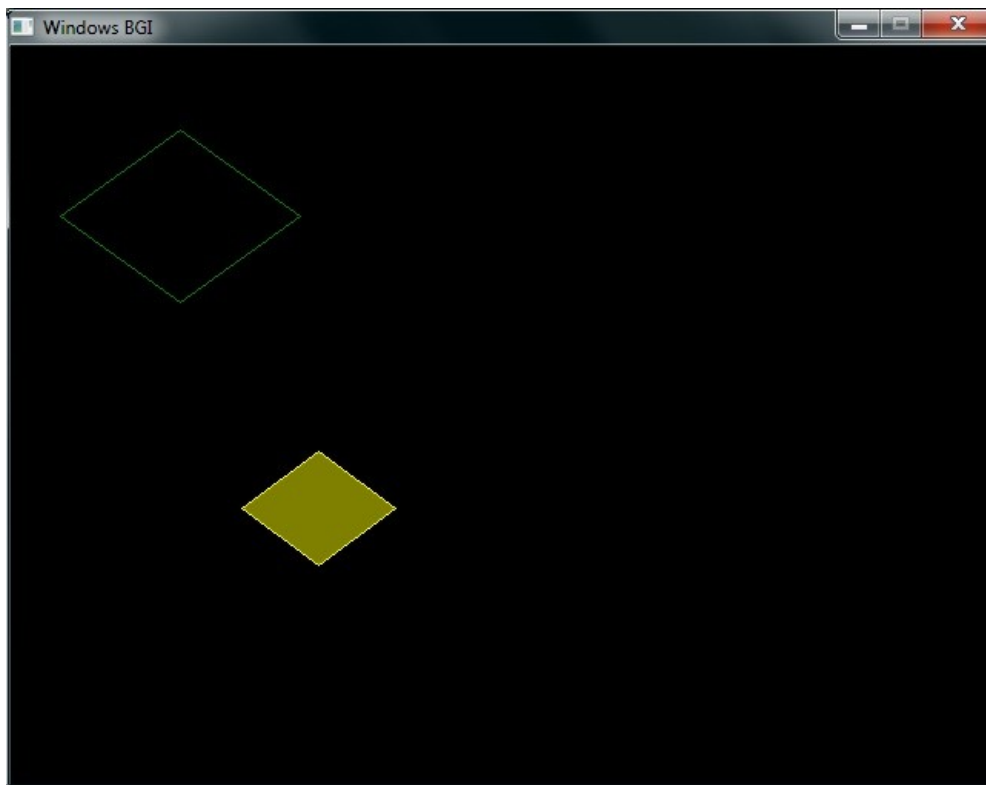


```

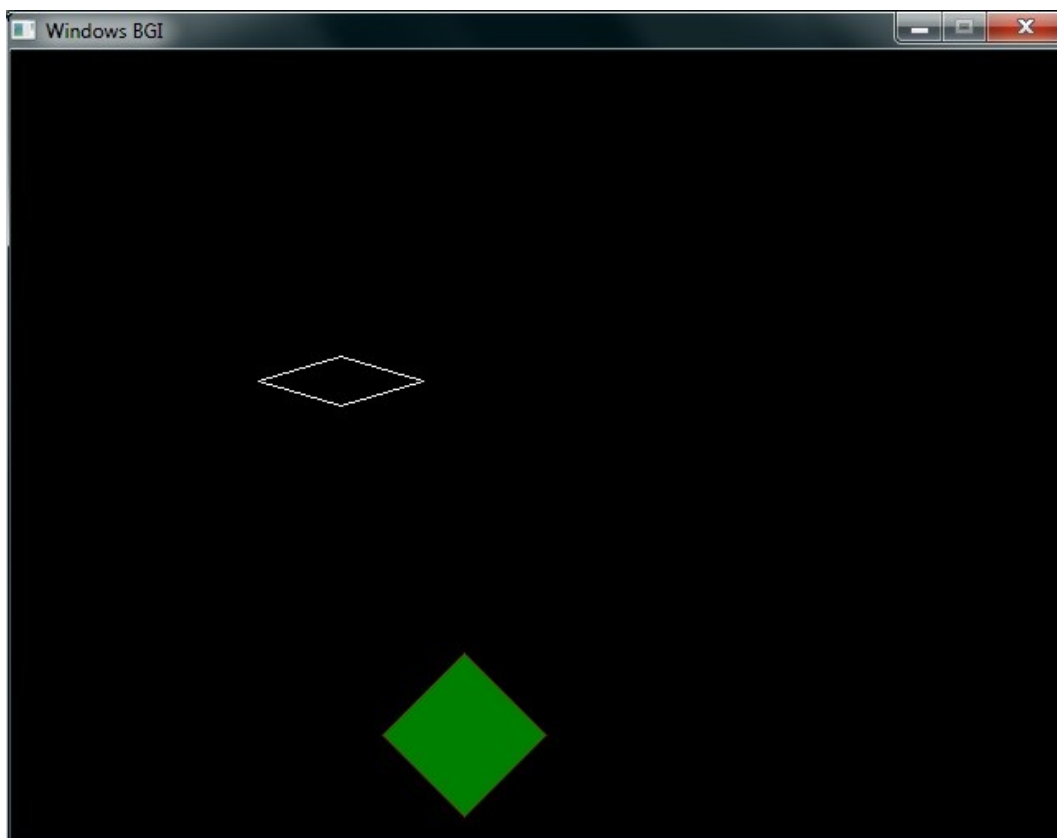
b->show();
getch();
a->hide();
b->hide();
getch();
a->move(90, 90);
b->move(75, 115);
a->show();
b->show();
getch();
a->setcolor(WHITE);
b->setcolor(RED);
getch();
// проверяем изменение размеров, обе фигуры меняются
if(Romb *r=dynamic_cast<Romb*>(a)) r->setsizes(100,30);
if(Romb *r=dynamic_cast<Romb*>(b)) r->setsizes(100,100);
getch();
// проверяем перекраску, фигура а не должна измениться
if(FillRomb *r=dynamic_cast<FillRomb*>(a)) r->setfillcolor(GREEN);
if(FillRomb *r=dynamic_cast<FillRomb*>(b)) r->setfillcolor(GREEN);
getch();
// проверяем исчезновение с экрана при удалении
delete a;
delete b;
getch();
return 0;
}

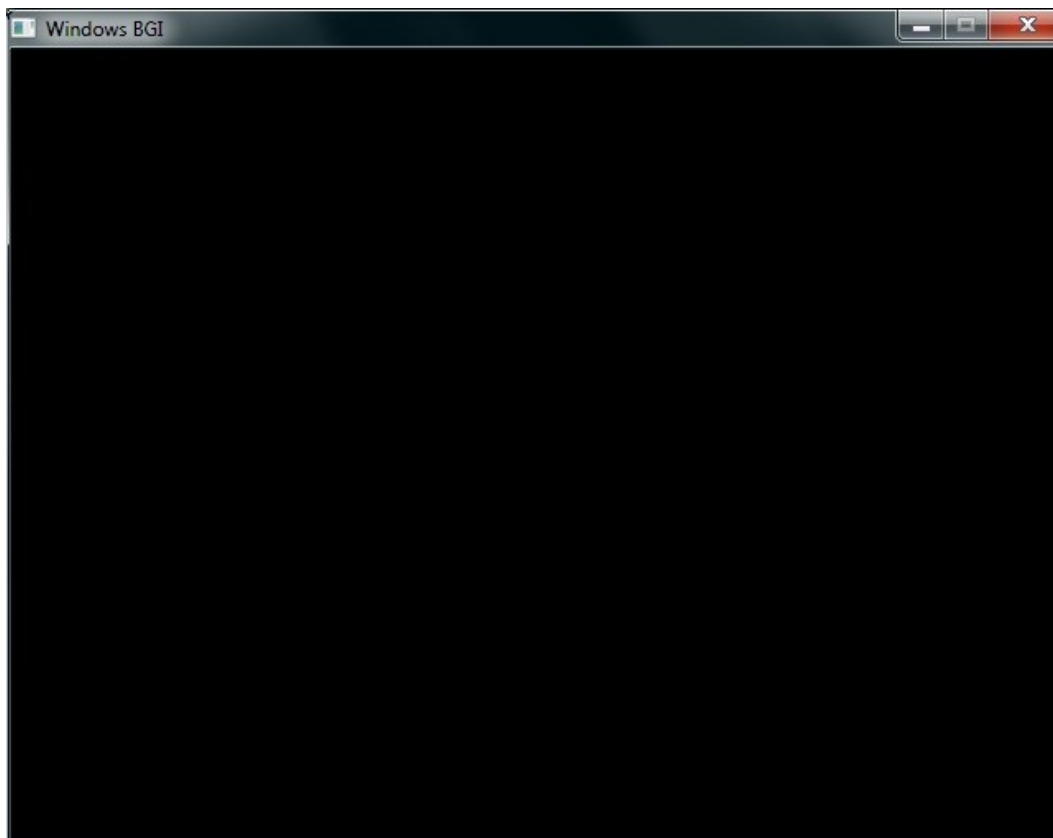
```

Полученные результаты:









4 Листинг реализации класса

```
void Figure::setcolor(int c) {
    this->c=c;
    if (visible) draw();
}
void Figure::move(int x, int y) {
    bool f=visible;
    if (f) hide();
    this->x=x;
    this->y=y;
    if (f) show();
}
void Figure::hide() {
    if (visible==0) return;
    int x1, y1, x2, y2;
    area(x1, y1, x2, y2);
    setfillstyle(SOLID_FILL, BLACK);
    bar(x1, y1, x2, y2);
    visible=0;
}
void Figure::show() {
    if (visible==1) return;
    visible=1;
    draw();
}
void Romb::setsizes(int l, int h) {
    bool f=isvisible();
```

```

    if (f) hide();
    this->l=l;
    this->h=h;
    if (f) show();
}
void Romb::area(int &x1, int &y1, int &x2, int &y2)const {
    x1=x-l/2;
    y1=y-h/2;
    x2=x+l/2;
    y2=y+h/2;
}
void Romb::draw() {
    ::setcolor(getcolor());
    moveto(x-l/2, y);
    lineto(x,y-h/2);
...
    lineto(x-l/2, y);
}
void FillRomb::draw() {
    int b[8]={x-l/2, y,
...
    x,y+h/2
    };
    setfillstyle(SOLID_FILL, fc);
    fillpoly(4, b);
    Romb::draw();
}
void FillRomb::setfillcolor(int a) {
    fc=a;
    if (isvisible()) draw();
}

```