ФГАОУВО «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-211

_____ М.В.Савонин

_____2022 г.

Работа зачтена с оценкой

_____

_____А.К.Демидов

_____2022 г.

Челябинск, 2022

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 точку (x,y)
              // 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() const; // получить цвет
  void hide();    // спрятать: нарисовать черный прямоугольник
                  //           по размерам area()
  void show();    // показать
  bool isvisible() const; // видима?
  virtual void area(int &x1,int &y1,int &x2,int &y2) const;
                // 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

12. Пятиугольная звезда
Pentagram(цвет линий, x и y центра, радиус, угол поворота)

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 конструтора, 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 классов

```
   class Figure
   {
      private:
         int c; // цвет
         bool visible; // видимость
      protected:
         int x,y; // базовая точка
         virtual void draw(); // нарисовать
      public:
         Figure(int c, int x, int y): c(c), x(x),
y(y){visible = false;} // Конструктор
         virtual ~Figure() {} // Деструктор
         void move(int x, int y); // переместить фигуру в
точку
         void setcolor(int c); // установить цвет фигуры
         int getcolor() const { return c; } // получить цвет
         void hide(); // спрятать фигуру
         void show(); // показать фигуру
         bool isvisible() const { return visible; } //
видима?
         virtual void area(int &x1, int &y1, int &x2, int
&y2) const = 0;
      // получить размеры прямоугольной области, содержащей
фигуру
   };

   class Pentagram: public Figure
   {
      protected:
         double r, fi; // длинна и высота стрелки
         void draw(); // нарисовать
      public:
         Pentagram(int c, int x, int y, int r, int fi):
Figure(c, x, y), r(r), fi(fi) {} // конструктор
         ~Pentagram(){hide();} // деструктор
         void setsizes(double r, double fi);// изменение
размера
         void area(int &x1,int &y1,int &x2,int &y2) const; //
получить размеры прямоугольной области, содержащей фигуру
   };

   class FillPentagram: public Pentagram
```

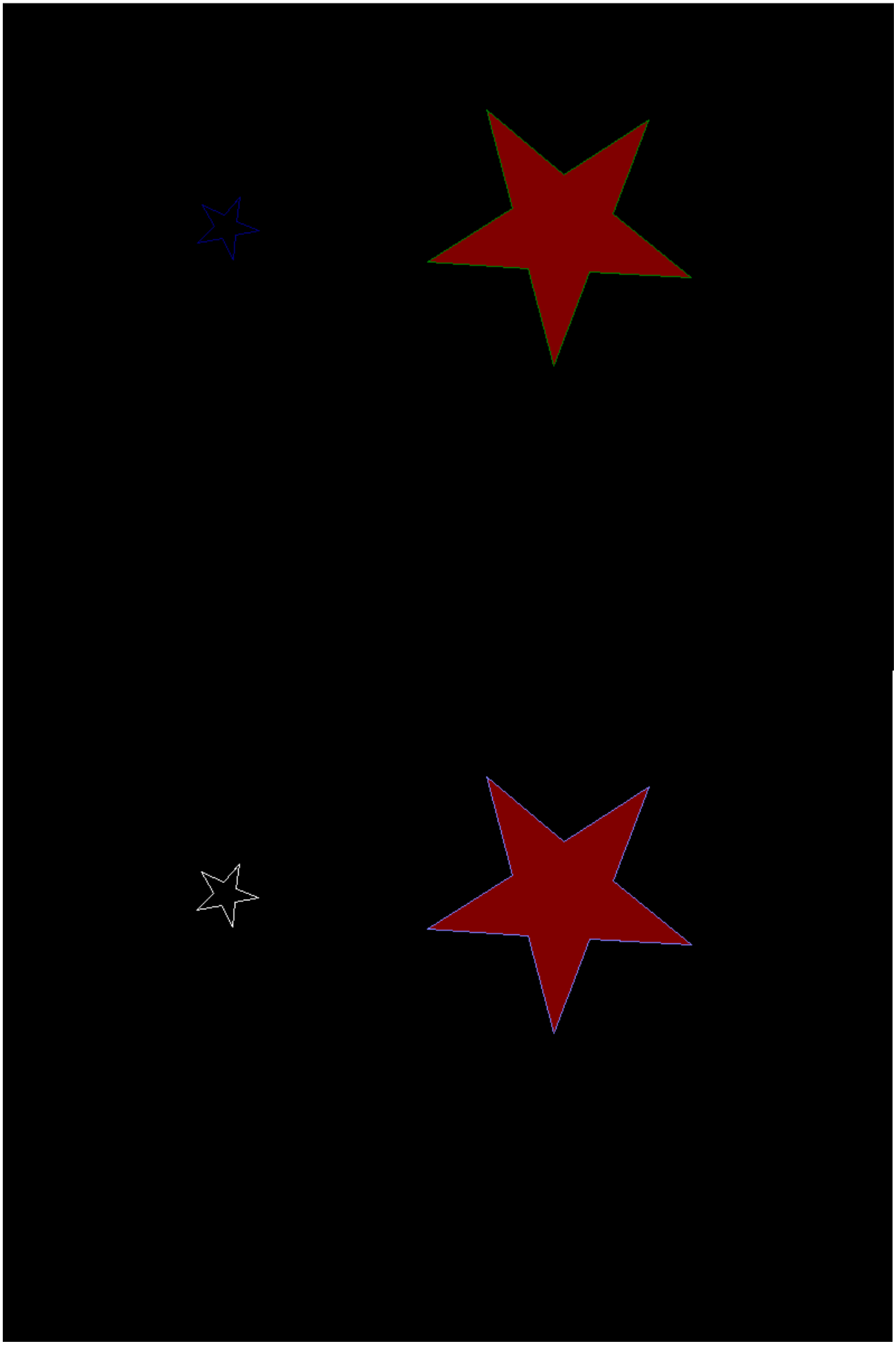
```
   {
      private:
         void fill(int x, int y); // закраска
         int fillColor; // цвет закраски
         void draw(); // нарисовать
      public:
         FillPentagram(int c, int x, int y, int r, int fi,
int fillColor): Pentagram(c, x, y, r, fi),
fillColor(fillColor){} //конструктор
         void setfillcolor(int c); // изменить цвет закраски
   };
```

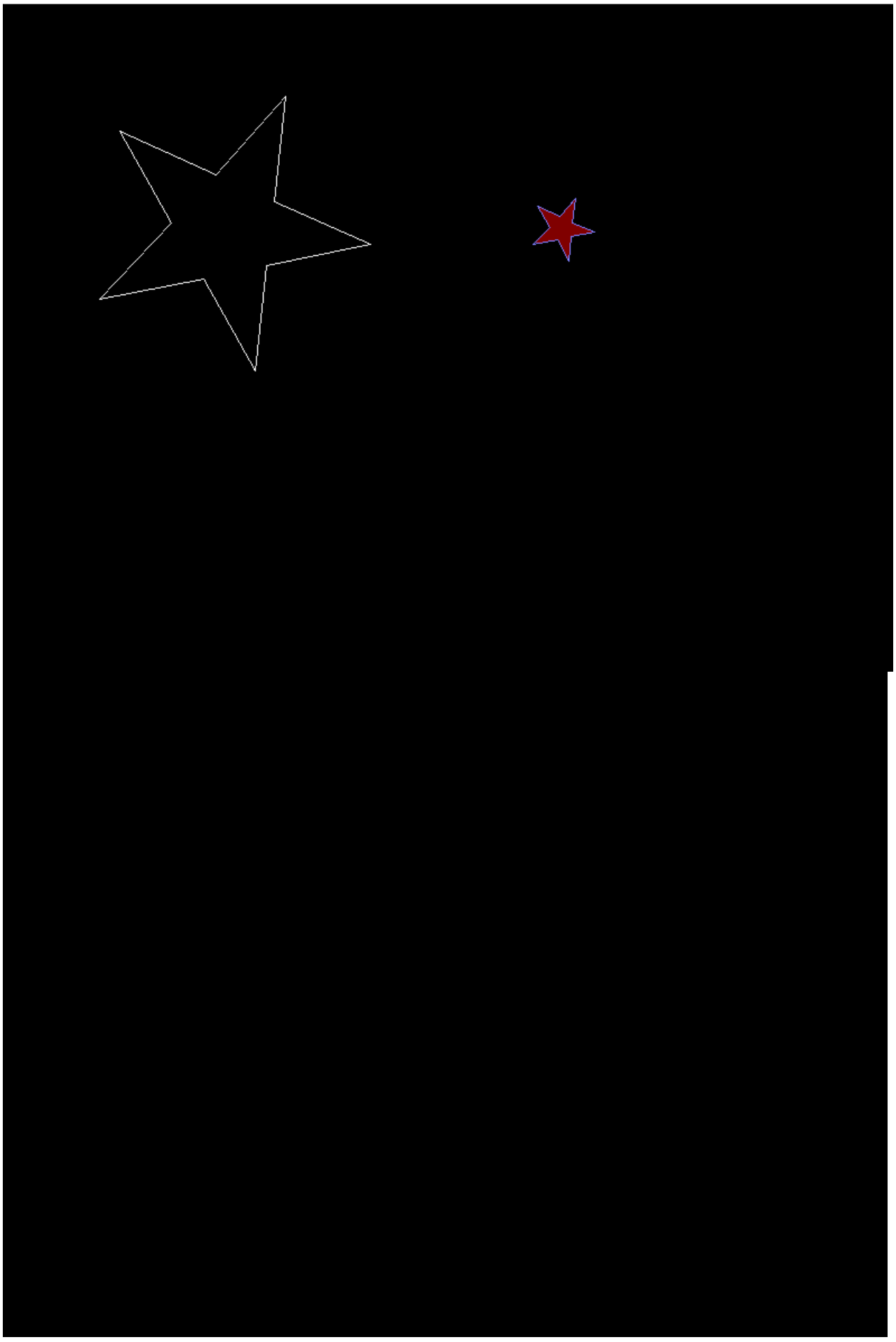
3 Описание тестов для проверки классов

```
   int main() {
      initwindow(800, 600);
      Figure *a=new Pentagram(BLUE, 100, 100, 30, 2);
      Figure *b=new FillPentagram(GREEN, 300, 100, 125, 1,
RED);
      a->show();
      b->show();
      while(!mousebuttons());
      while(mousebuttons());
      a->hide();
      b->hide();
      while(!mousebuttons());
      while(mousebuttons());
      a->move(200, 200);
      b->move(500, 200);
      a->show();
      b->show();
      while(!mousebuttons());
      while(mousebuttons());
      a->setcolor(WHITE);
      b->setcolor(LIGHTBLUE);
      while(!mousebuttons());
      while(mousebuttons());
   // проверяем изменение размеров, обе фигуры меняются
      if (Pentagram *r=dynamic_cast<Pentagram *>(a)) r-
>setsizes(130, 2);
      if (Pentagram *r=dynamic_cast<Pentagram *>(b)) r-
>setsizes(30, 2);
      while(!mousebuttons());
      while(mousebuttons());
   // проверяем перекраску, фигура a не должна измениться
      if (FillPentagram *r=dynamic_cast<FillPentagram *>(a))
r->setfillcolor(GREEN);
```

```
      if (FillPentagram *r=dynamic_cast<FillPentagram *>(b))
r->setfillcolor(GREEN);
      while(!mousebuttons());
      while(mousebuttons());
   // проверяем исчезновение с экрана при удалении
      delete a;
      delete b;
      while(!mousebuttons());
      while(mousebuttons());
      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 s = visible;
      if (s) hide();
      this->x = x;
      this->y = y;
      if (s) 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) return;
      draw();
      visible = 1;
   }

   void Pentagram::area(int &x1,int &y1,int &x2,int &y2)
const {
      x1 = x-r;
      y1 = y-r;
      x2 = x+r;
      y2 = y+r;
   }

   void Pentagram::setsizes(double r, double fi) {
      bool s = isvisible();
      if (s) hide();
      this->r = r;
      this->fi = fi;
      if (s) show();
   }
```

```
   void Pentagram::draw() {
      ::setcolor(getcolor());
      int x0, y0;
      x0 = x+int(r*(cos(3.14/5*2)/cos(3.14/5))*cos(fi));
      y0 = y+int(r*(cos(3.14/5*2)/cos(3.14/5))*sin(fi));
      for(int i = 1; i < 11; i++)
      {
         int y1, x1;
         if(i%2)
         {
            x1 = x+int(r*cos(i*3.14/5+fi));
            y1 = y+int(r*sin(i*3.14/5+fi));
         }
         else
         {
            x1 =
x+int(r*(cos(3.14/5*2)/cos(3.14/5))*cos(i*3.14/5+fi));
            y1 =
y+int(r*(cos(3.14/5*2)/cos(3.14/5))*sin(i*3.14/5+fi));
         }
         line(x0, y0, x1, y1);
         x0 = x1;
         y0 = y1;
      }
   }

   void FillPentagram::fill(int x, int y)
   {
      if(getpixel(x, y) == 0)
      {
         putpixel(x, y, fillColor);
         fill(x+1, y);
         fill(x-1, y);
         fill(x, y+1);
         fill(x, y-1);
      }
   }

   void FillPentagram::draw() {
      setfillstyle(SOLID_FILL, fillColor);
      Pentagram::draw();
      fill(x, y);
   }

   void FillPentagram::setfillcolor(int c) {
      fillColor = c;
```

```
    if (isvisible()) draw();
}
```