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

_____ В.А.Малахов

_____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

6. Сектор круга (незакрашенный)
Sector(цвет линий, x и y цетра круга, радиус, угол1, угол2)

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
   {
      int c; // цвет
      bool visible; // видимость
      protected:
         int x,y; // базовая точка
         virtual void draw(); // нарисовать
      public:
         Figure(int x, int y, int c): x(x), y(y),
c(c){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 Sector: public Figure
   {
      protected:
         double r, fi1, fi2; // длинна и высота стрелки
         void draw(); // нарисовать
      public:
         Sector(int x, int y, int c, double r, double fi1,
double fi2): Figure(x, y, c), r(r), fi1(fi1), fi2(fi2) {} //
конструктор
         ~Sector(){hide();} // деструктор
         void setsizes(double r, double fi1, double fi2);//
изменение размера
         void area(int &x1,int &y1,int &x2,int &y2) const; //
получить размеры прямоугольной области, содержащей фигуру
   };

   class FillSector: public Sector
```

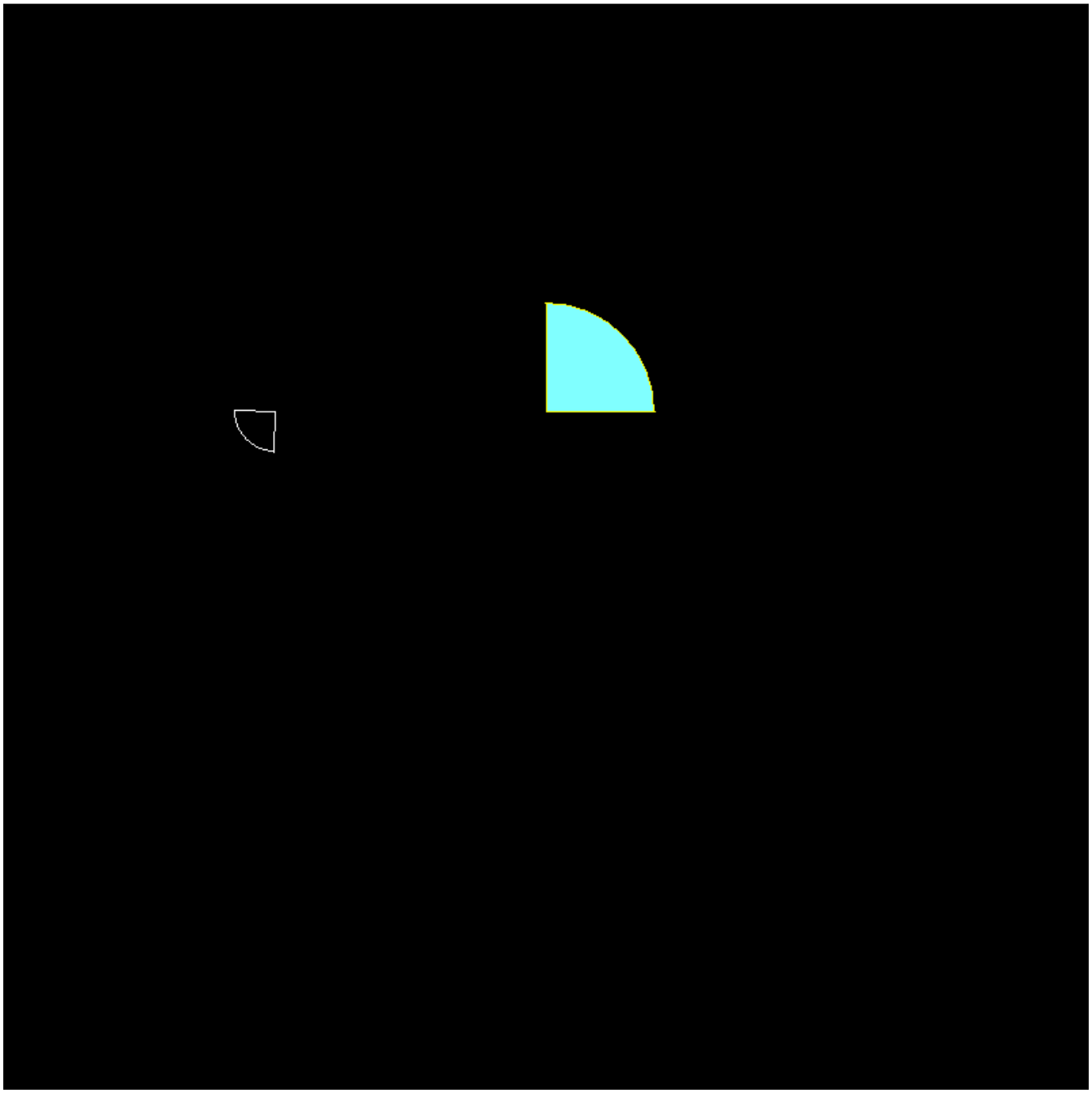
```
   {
      int fillc; // цвет закраски
      void draw(); // нарисовать
      public:
         FillSector(int x, int y, int c, double r, double
fi1, double fi2, int fillc): Sector(x, y, c, r, fi1, fi2),
fillc(fillc){} //конструктор
         void setfillcolor(int c); // изменить цвет закраски
   };
```

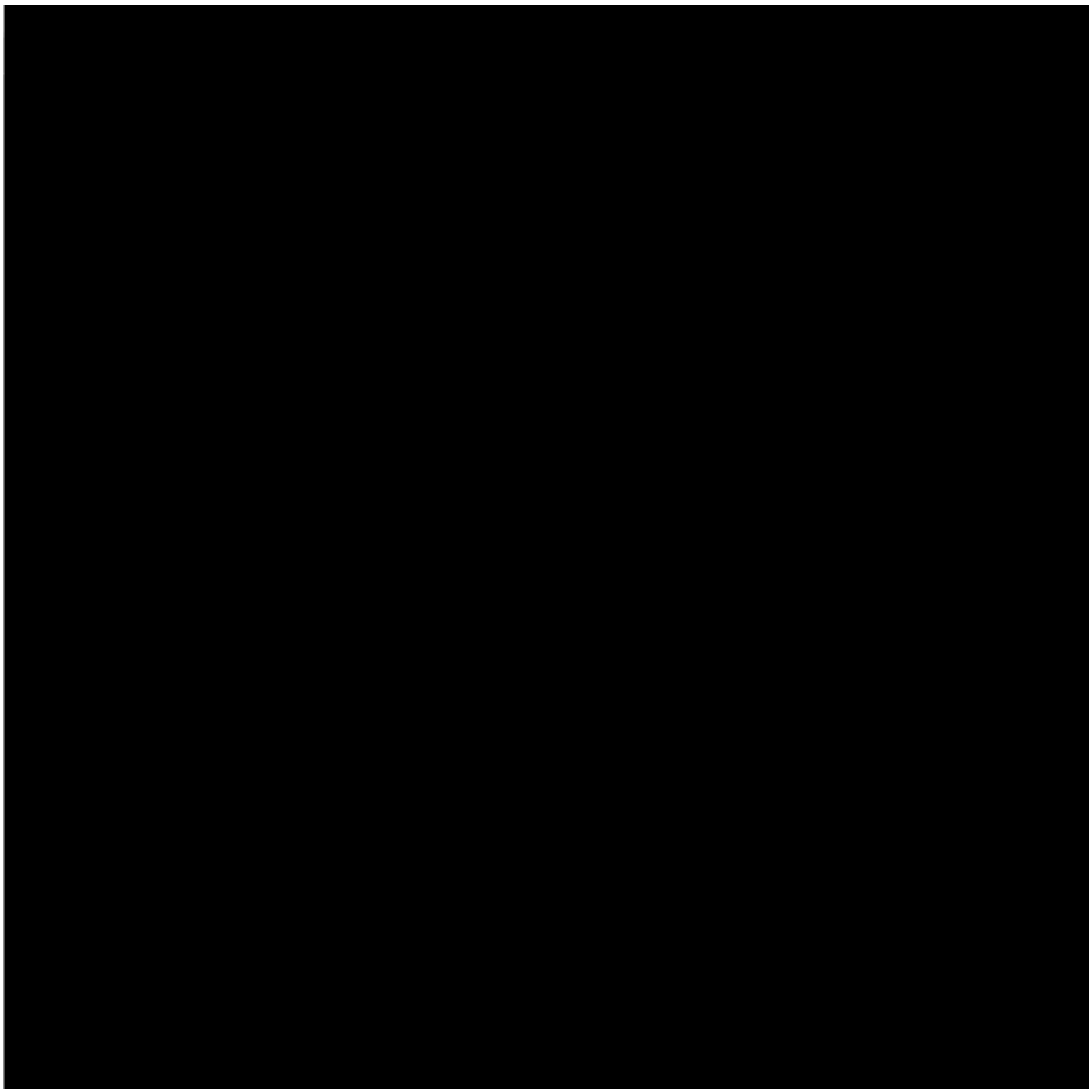
3 Описание тестов для проверки классов

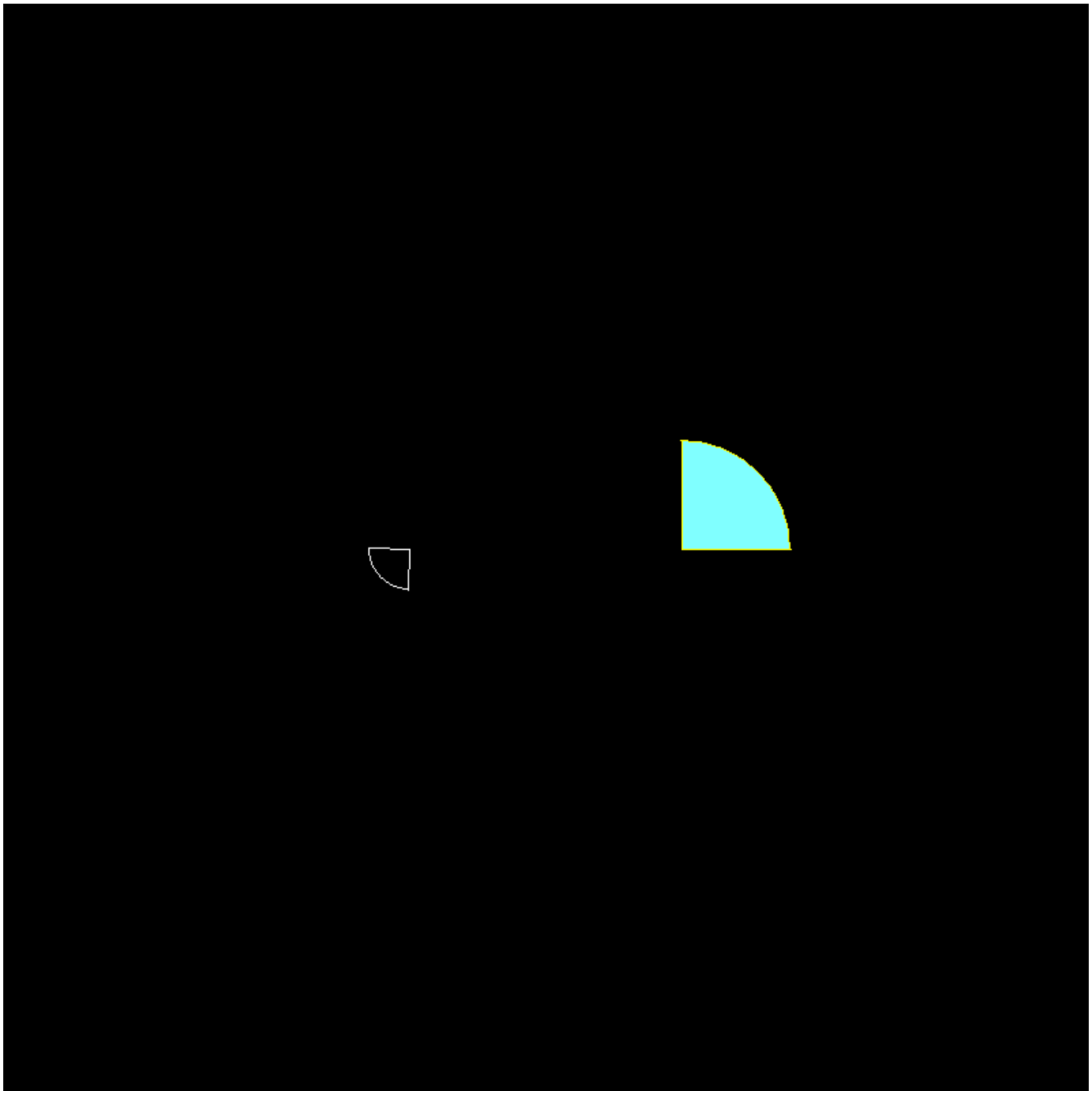
```
   int main() {
      initwindow(800, 800);
      Figure *a=new Sector(200, 300, WHITE, 30, 180, 270);
      Figure *b=new FillSector(400, 300, YELLOW, 80, 0, 90,
LIGHTCYAN);

      a->show();
      b->show();
      getch();
      a->hide();
      b->hide();
      getch();
      a->move(300, 400);
      b->move(500, 400);
      a->show();
      b->show();
      getch();
      a->setcolor(GREEN);
      b->setcolor(LIGHTBLUE);
      getch();
   // проверяем изменение размеров, обе фигуры меняются
      if (Sector *r=dynamic_cast<Sector *>(a)) r-
>setsizes(40, 100, 180);
      if (Sector *r=dynamic_cast<Sector *>(b)) r-
>setsizes(100, 30, 80);
      getch();
   // проверяем перекраску, фигура a не должна измениться
      if (FillSector *r=dynamic_cast<FillSector *>(a)) r-
>setfillcolor(GREEN);
      if (FillSector *r=dynamic_cast<FillSector *>(b)) r-
>setfillcolor(GREEN);
      getch();
   // проверяем исчезновение с экрана при удалении
      delete a;
      delete b;
      getch();
```

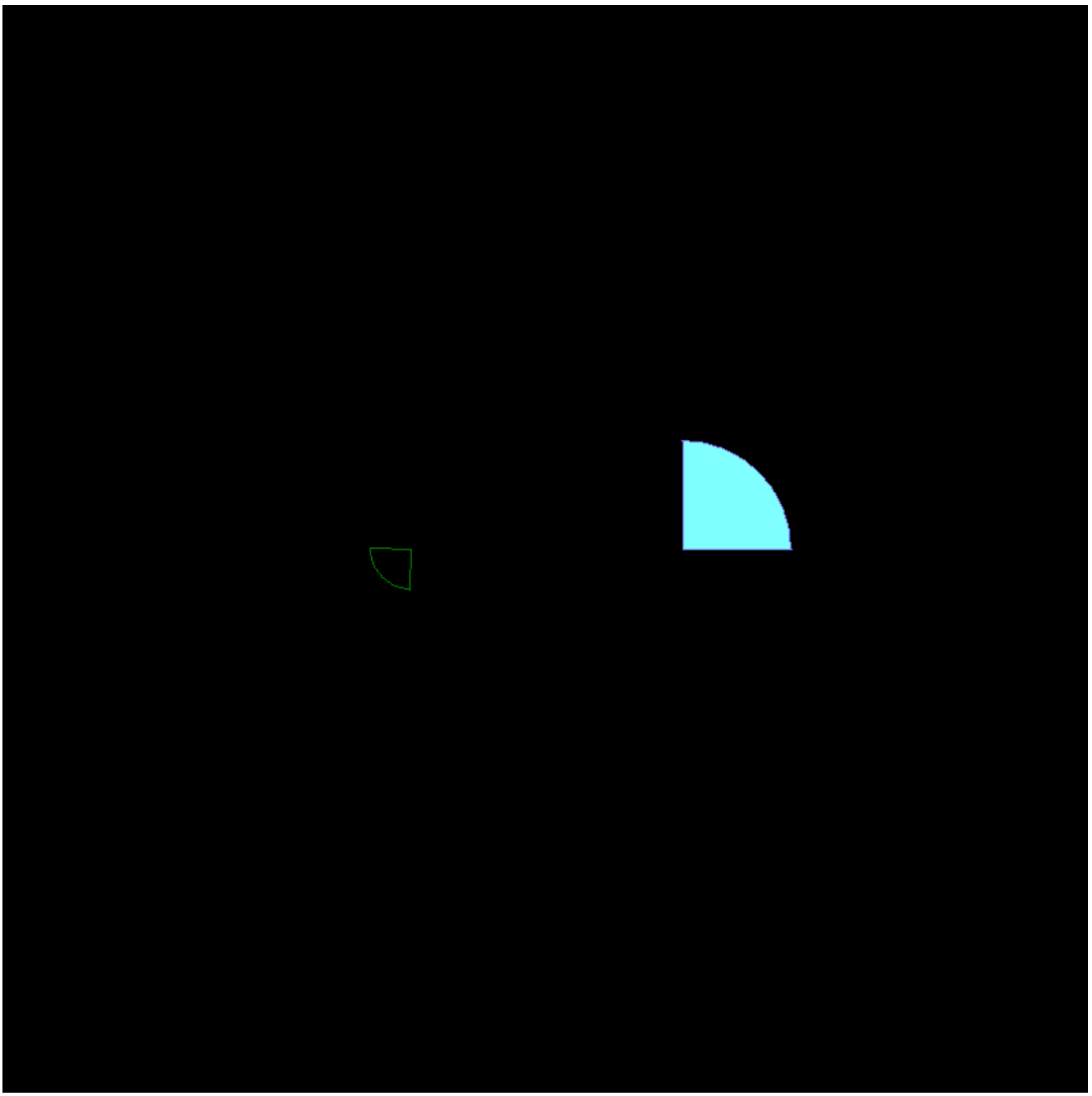
```
    return 0;
}
```

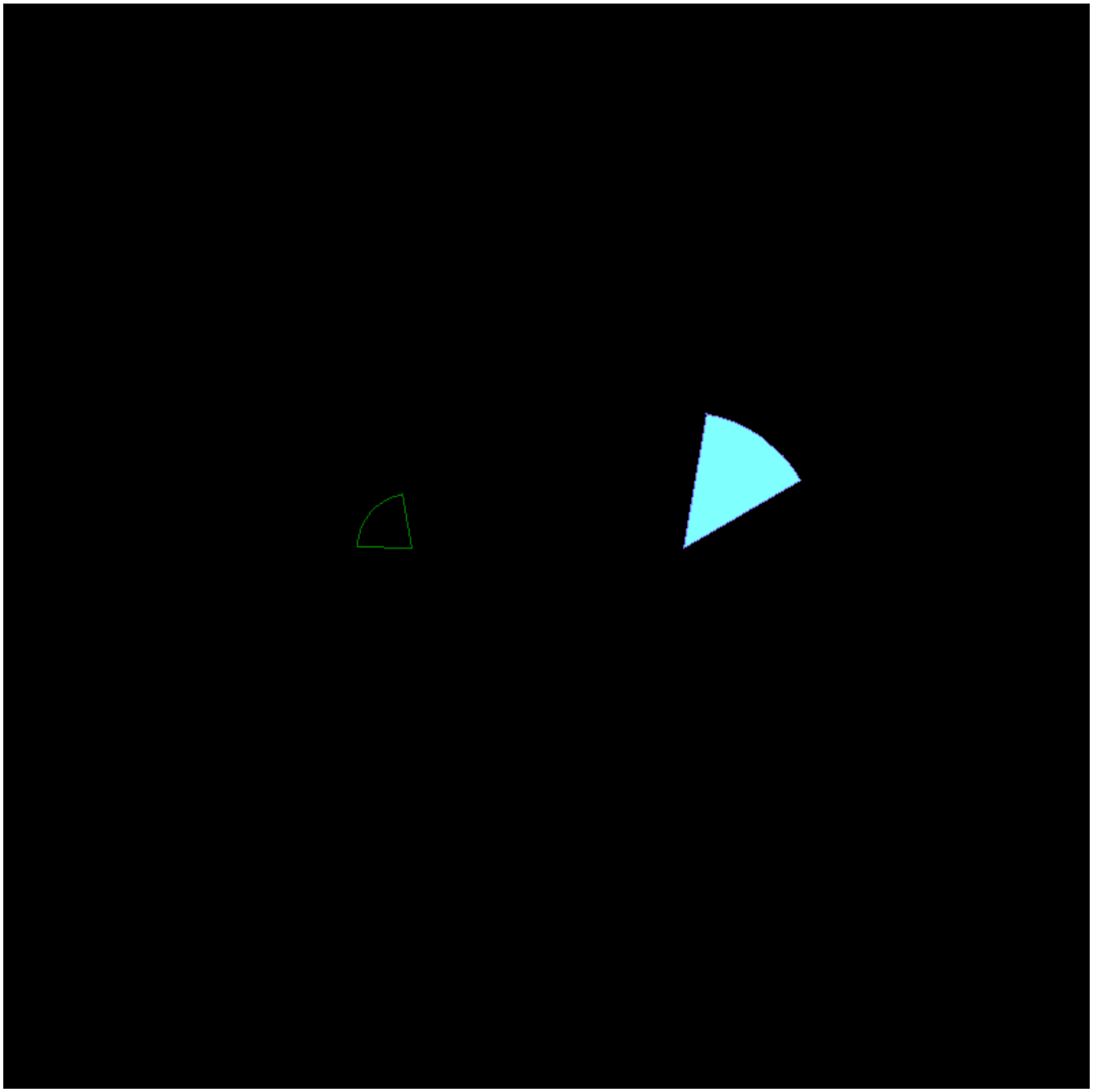
Полученные результаты

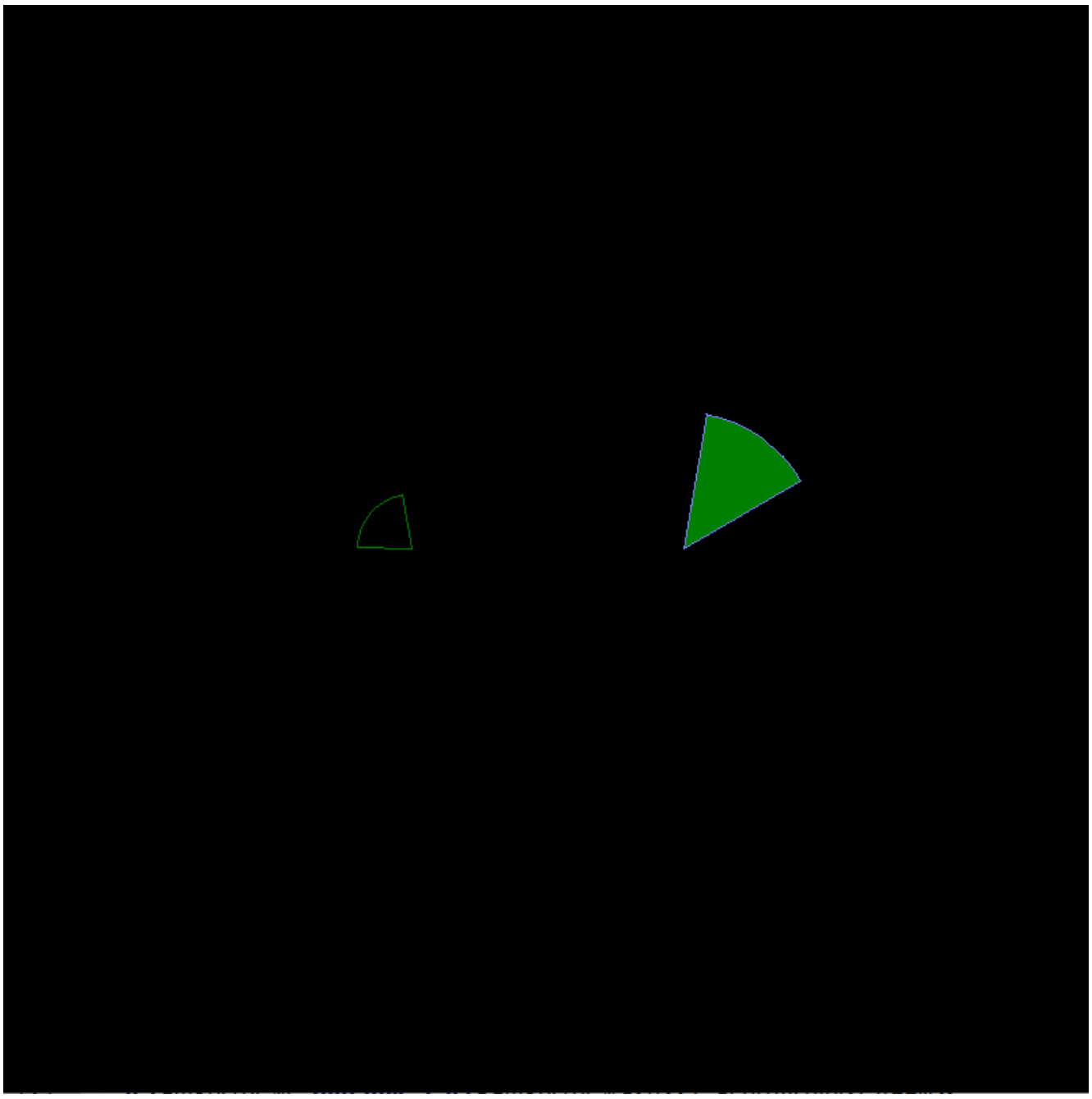


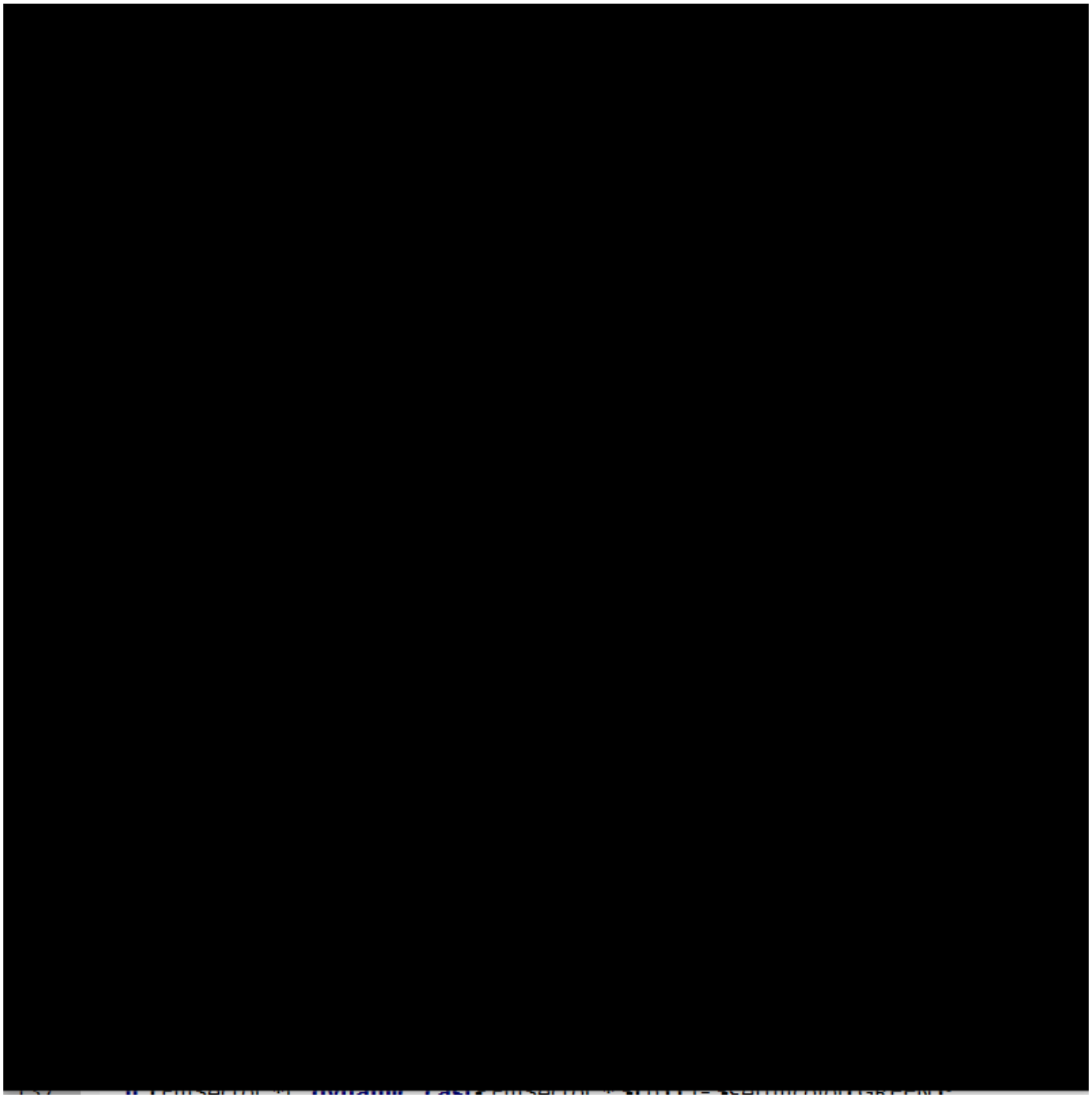












4 Листинг реализации класса

```
void Figure::setcolor(int c) {
   this->c = c;
   if (visible) draw();
}

void Figure::move(int x, int y) {
   bool 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 Sector::area(int &x1,int &y1,int &x2,int &y2) const {
   x1 = x+fmin(0, fmin(r*cos(fi1/180.0*M_PI),
r*cos(fi2/180.0*M_PI)))-1;
   y1 = y+fmin(0, fmin(-r*sin(fi1/180.0*M_PI), -
r*sin(fi2/180.0*M_PI)))-1;
   x2 = x+fmax(0, fmax(r*cos(fi1/180.0*M_PI),
r*cos(fi2/180.0*M_PI)))+1;
   y2 = y+fmax(0, fmax(-r*sin(fi1/180.0*M_PI), -
r*sin(fi2/180.0*M_PI)))+1;
}

void Sector::setsizes(double r, double fi1, double fi2) {
   bool s = isvisible();
   if (s) hide();
   this->r = r;
   this->fi1 = fi1;
   this->fi2 = fi2;
   if (s) show();
}

void Sector::draw() {
   ::setcolor(getcolor());
   arc(x, y, fi1, fi2, r);
   line(x, y, x+r*cos(fi1/180.0*M_PI), y-
r*sin(fi1/180.0*M_PI));
   line(x, y, x+r*cos(fi2/180.0*M_PI), y-
r*sin(fi2/180.0*M_PI));
}

void FillSector::draw() {
```

```
   setfillstyle(SOLID_FILL, fillc);
   Sector::draw();
   int x1, x2, y1, y2;
   Sector::area(x1, y1, x2, y2);
   floodfill((x1+x2)/2, (y1+y2)/2, getcolor());
}
void FillSector::setfillcolor(int c) {
   fillc = c;
   if (isvisible()) draw();
}
```