Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

О Т Ч Е Т

по домашней работе № 1

Название: Программирование на Object Pascal с использованием классов

Дисциплина: Объектно-ориентированное программирование

Студент ИУ6-24 Б _____ М.К. Цыпленков

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____

(Подпись, дата) (И.О. Фамилия)

Москва, 2022

Часть 1.1. Графика. Обработка события нажатия клавиши мыши. Наследование. Полиморфизм

Условие задания: Разработать иерархию классов. Поместить определение классов в отдельном модуле. Разработать программу, содержащую описание трех графических объектов:

правильный треугольник, треугольная пирамида (прозрачная), треугольная призма (прозрачная).

Реализуя механизм полиморфизма, привести объекты в синхронное вращение вокруг их вертикальных осей. Скорость вращения регулируется с использованием интерфейсных элементов.

В отчете показать иерархии используемых классов VCL и разработанных классов, граф состояния пользовательского интерфейса и объектную декомпозицию.

Решение

1) Как задавать фигуры (по координатам или другими способами).

Задавать фигуры координатами вершин в рамках поставленной задачи не удобно, так как они будут вращаться, и будет очень трудно рассчитать траекторию движения вершин, учитывая все координаты. Так как в условии сказано, что треугольник, пирамида и призма – правильные, то треугольник будем задавать длиной стороны, а пирамиду и призму – длиной основания и высотой.

2) Преобразование координат.

Условимся, что треугольник, основание пирамиды и призмы лежат в одной плоскости: xOy (таким образом, все соответствующие вершины имеют координату z = 0). Также условимся, что треугольник, основание пирамиды и призмы занимают положение (в начальный момент времени), приведенное на рисунке 1.1.

a – сторона треугольника (или сторона основания пирамиды или призмы).

C (Xc; Yc) – координаты центра треугольника (основания пирамиды или призмы).

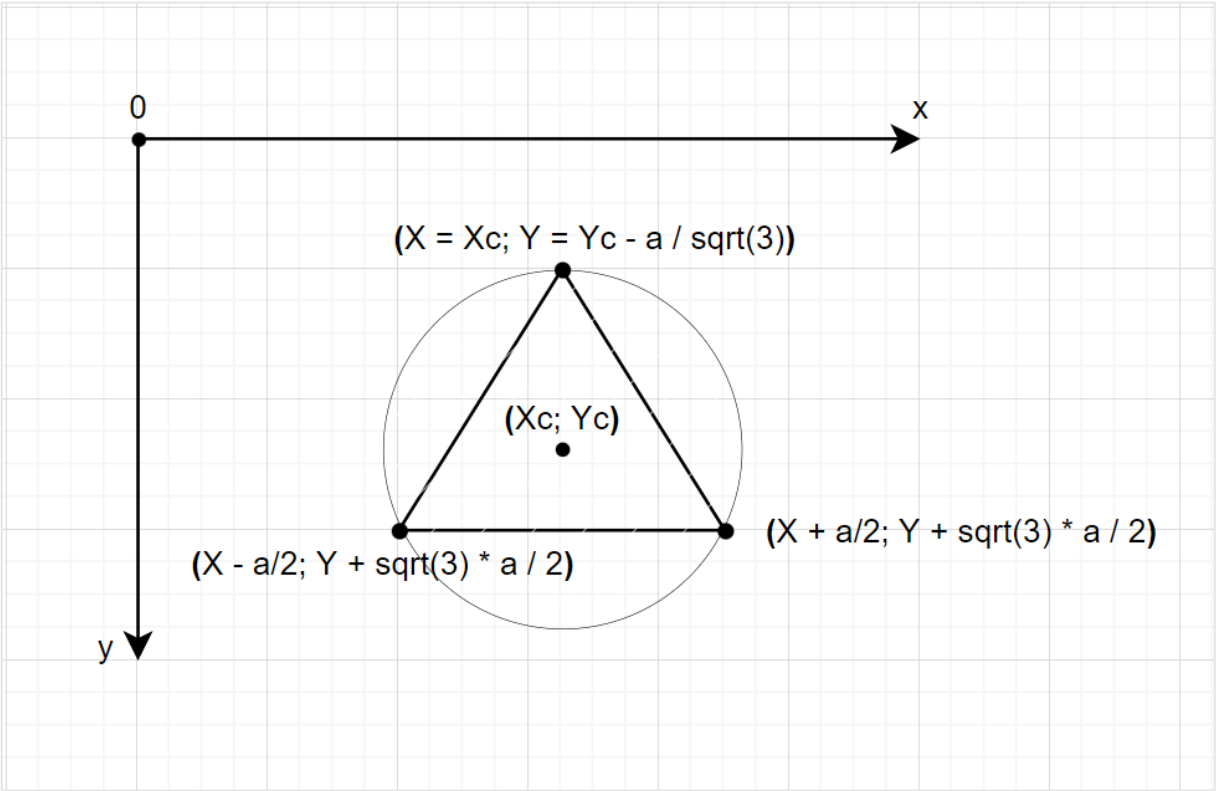

Рисунок 1.1 – начальное положение треугольника
(основания пирамиды или призмы)

Очевидно, что точка C является проекцией оси вращения для треугольника, пирамиды и призмы. Поэтому реализуем функции для преобразования координат точки при ее повороте вокруг этой точки (см рисунок 1.2).

Высота пирамиды и призмы будет учитываться непосредственно в методах построения их изображения.

```
    {преобразование координат}
    function new_x(x, y, x_c, y_c, t : real) : integer;
55  begin
      new_x := round((x - x_c)*cos(t) + (y - y_c)*sin(t) + x_c);
    end;

    function new_y(x, y, x_c, y_c, t, phi : real) : integer;
60      begin
      new_y := round(((y - y_c)*cos(t) - (x - x_c)*sin(t) + y_c)
    end;
```

Рисунок 1.2 – коды функций для расчета координат новой точки,
образованной поворотом одной точки вокруг другой

Треугольник – плоская фигура, поэтому его представление в пространстве можно свести к плоскости. Но у нас также присутствуют трехмерные фигуры. Для их наглядного изображения была выбрана плоскость, параллельная оси Ox и пересекающая плоскость xOy под углом α ($0^\circ < \alpha < 90^\circ$). Таким образом, на эту плоскость координаты X проецируются без искажений, а координаты Y будут рассчитываться по следующей формуле: $y' = y * \cos(\alpha) - z * \sin(\alpha)$.

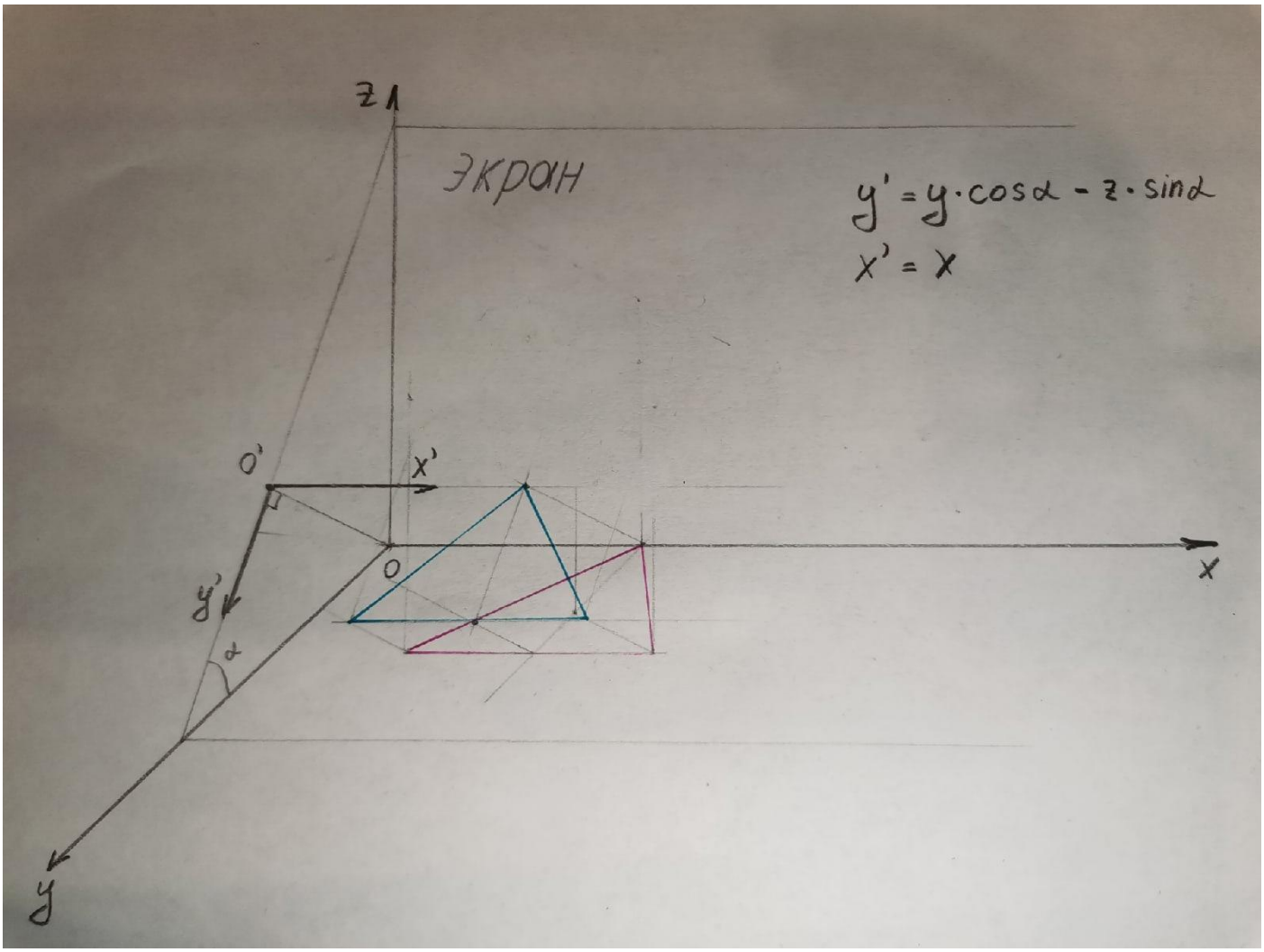

Рисунок 1.3 – наглядное изображение замены плоскостей

У треугольника, как говорилось раньше, координаты вершин по Z равны 0, как и у оснований пирамиды и призмы, поэтому $y' = y * \cos(\alpha)$, а высоту пирамиды и призмы учтем непосредственно в методах построения их изображения.

3) На рисунках ниже представлен код модуля unit2, в котором реализованы классы для представления правильного треугольника, пирамиды и призмы, и методы для работы с ними, и остальные необходимые схемы и диаграммы.

Реализация классов

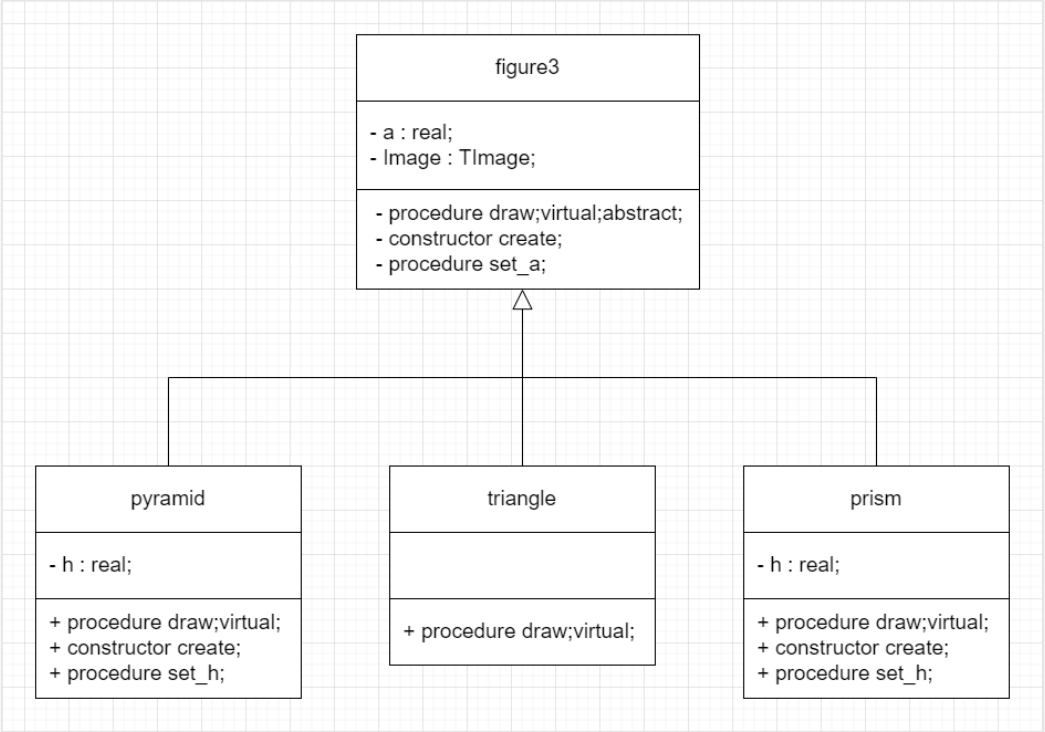

Рисунок 2.1 – диаграмма классов

```
8   {общий предок}
    type figure3 = object
10  private
        a : real;
        Image : TImage;
    public
      procedure draw(xc, yc, t, phi : real; axis : boolean);virtual;abstract;
15    constructor create(aa : real;Aimage:TImage);
      procedure set_a(new_a : real);

    end;

20  {треугольник}
    type triangle = object(figure3)
      private
      public
        procedure draw(xc, yc, t, phi : real; axis : boolean);virtual;
25  end;

    {пирамида}
    type pyramid = object(figure3)
      private
30      h : real;
      public
        procedure draw(xc, yc, t, phi : real; axis : boolean);virtual;
        constructor create(aa : real; ah : real; Aimage:TImage);
        procedure set_h(new_h : real);
35  end;

    {призма}
    type prism = object(figure3)
      private
40      h : real;
      public
        procedure draw(xc, yc, t, phi : real; axis : boolean);virtual;
        constructor create(aa : real; ah : real; Aimage:TImage);
        procedure set_h(new_h : real);
45    end;
```

Рисунок 2.2 – программная реализация классов (поля классов)

```
    procedure figure3.set_a(new_a : real);
65  begin
      a := new_a;
    end;

    constructor figure3.create(aa : real; AImage:TImage);
70  begin
      a := aa;
      Image := Aimage;
      Image.Canvas.Pen.width := 4;
    end;
```

Рисунок 3.1 – методы общего предка

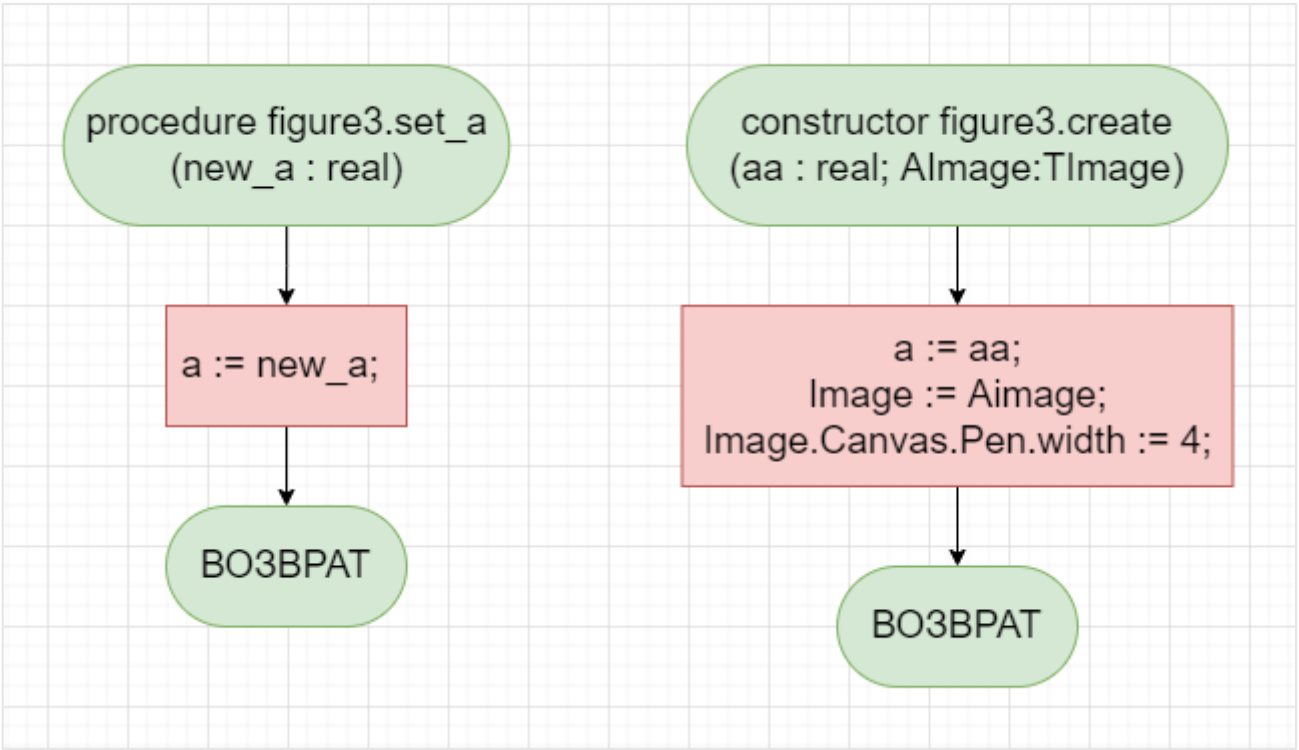

Рисунок 3.2 – схемы методов общего предка

Методы класса triangle

```
    procedure triangle.draw(xc, yc, t, phi : real; axis : boolean);
80  var x, y: real;
    begin
      x := xc;
      y := yc - a / sqrt(3);
      if axis then begin
85       Image.Canvas.Pen.color := clblack;
         Image.Canvas.MoveTo(round(xc), round(yc*cos(phi) - 30));
         Image.Canvas.Pen.Style := psDashDot;
         Image.Canvas.Pen.width := 2;
         Image.Canvas.LineTo(round(xc), round(yc*cos(phi) + 30));
90       Image.Canvas.Pen.Style := psSolid;
         Image.Canvas.Pen.width := 4;
      end;

      Image.Canvas.MoveTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi));
95
      Image.Canvas.Pen.color := clblue;
      Image.Canvas.LineTo(new_x(x + a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x + a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));

      Image.Canvas.Pen.color := clgreen;
100   Image.Canvas.LineTo(new_x(x - a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x - a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));

      Image.Canvas.Pen.color := clred;
      Image.Canvas.LineTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi));
    end;
```

Рисунок 4.1 – метод класса triangle

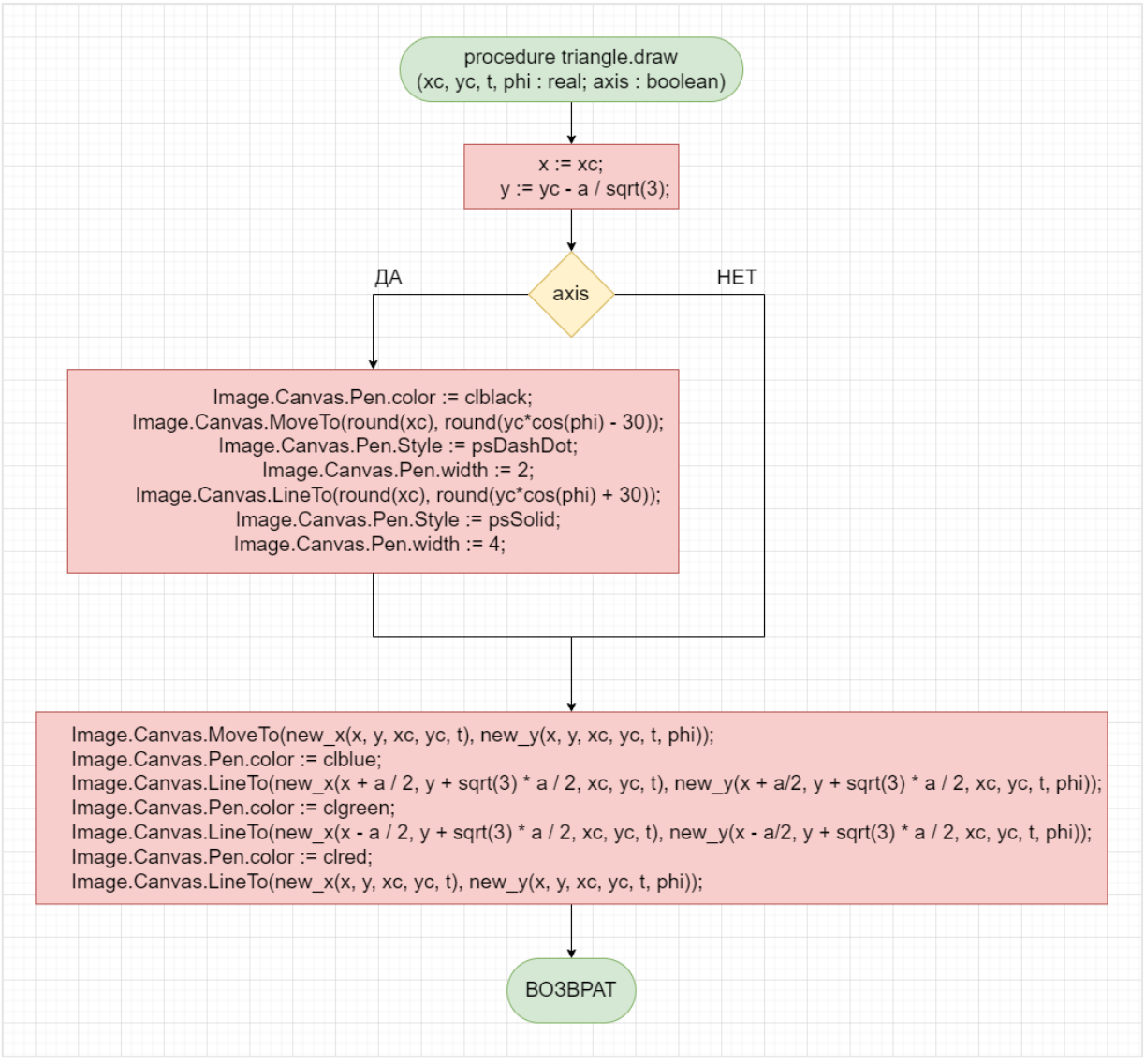

Рисунок 4.2 – схема метода draw

Методы класса pyramid

```
    procedure pyramid.set_h(new_h : real);
    begin
      h := new_h;
    end;

    constructor pyramid.create(aa : real; ah : real; Aimage:TImage);
    begin
      inherited create(aa, Aimage);
      h := ah;

    end;

    procedure pyramid.draw(xc, yc, t, phi : real; axis : boolean);
    var x, y, xpeak, ypeak : real;
    begin
      x := xc;
      y := yc - a / sqrt(3);
      xpeak := xc;
      ypeak := yc * cos(phi) - h * sin(phi);  {phi}
      if axis then begin
         Image.Canvas.Pen.color := clblack;
         Image.Canvas.MoveTo(round(xc), round(yc*cos(phi) + 30));
         Image.Canvas.Pen.Style := psDashDot;
         Image.Canvas.Pen.width := 2;
         Image.Canvas.LineTo(round(xc), round(ypeak - 30));
         Image.Canvas.Pen.Style := psSolid;
         Image.Canvas.Pen.width := 4;
      end;

      Image.Canvas.MoveTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi));

      Image.Canvas.Pen.color := clblue;
      Image.Canvas.LineTo(new_x(x + a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x + a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));

      Image.Canvas.Pen.color := clfuchsia;
      Image.Canvas.LineTo(round(xpeak), round(ypeak));

      Image.Canvas.MoveTo(new_x(x + a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x + a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));

      Image.Canvas.Pen.color := clgreen;
      Image.Canvas.LineTo(new_x(x - a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x - a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));

      Image.Canvas.Pen.color := claqua;
      Image.Canvas.LineTo(round(xpeak), round(ypeak));
      Image.Canvas.MoveTo(new_x(x - a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x - a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));

      Image.Canvas.Pen.color := clred;
      Image.Canvas.LineTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi));

      Image.Canvas.Pen.color := clpurple;
      Image.Canvas.LineTo(round(xpeak), round(ypeak));
    end;
```

Рисунок 5.1 – методы класса pyramid

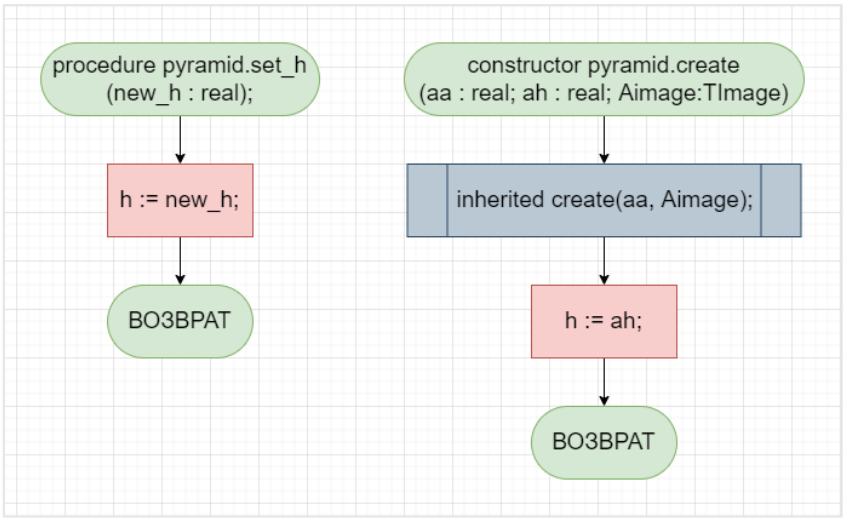

Рисунок 5.2 – схема методов pyramid (1)

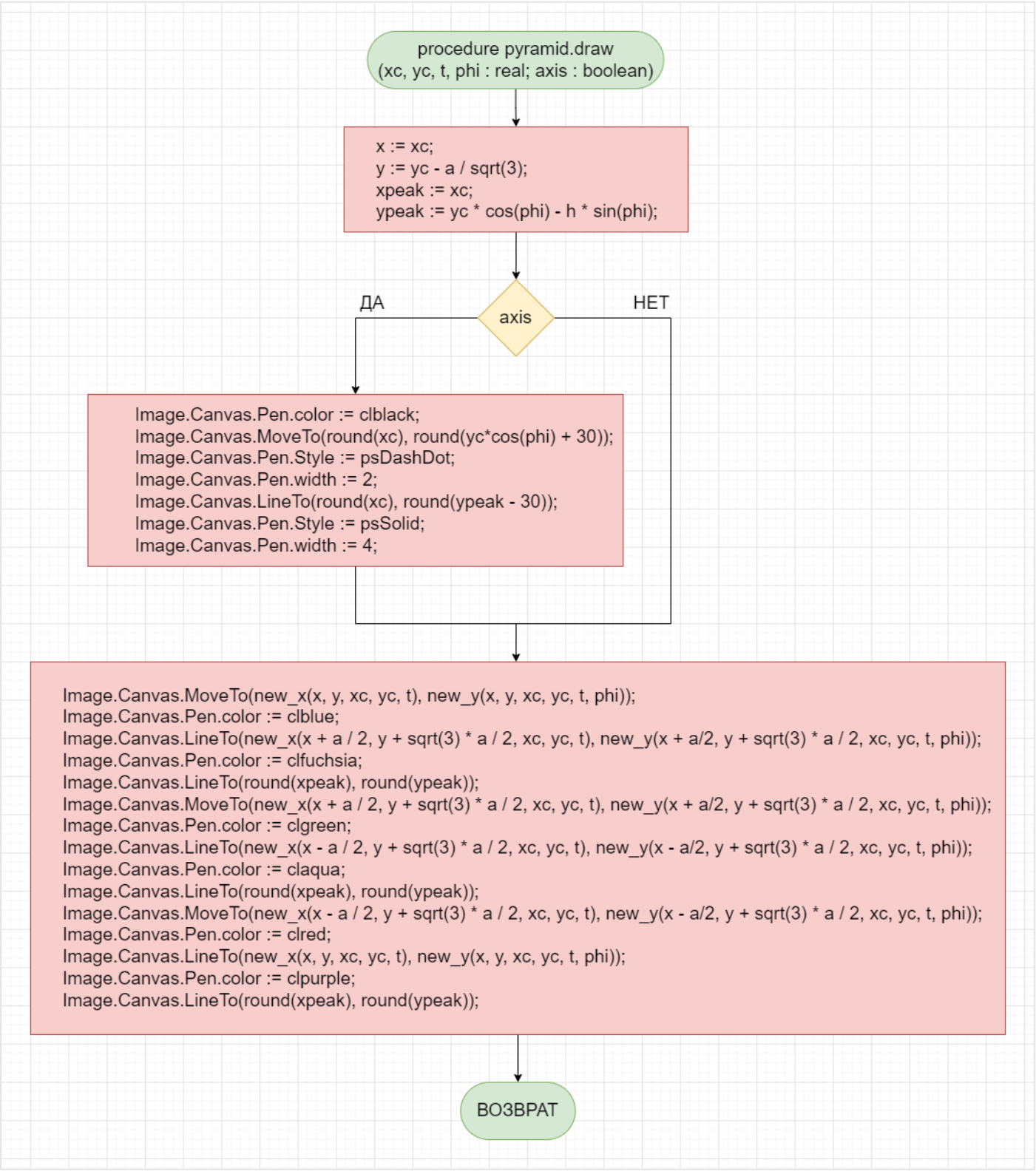

Рисунок 5.3 – схема методов pyramid (2)

Методы класса prism

```
    procedure prism.set_h(new_h : real);
    begin
      h := new_h;
    end;

    constructor prism.create(aa : real; ah : real; Aimage:TImage);
    begin
      inherited create(aa, Aimage);
      h := ah;

    end;

    procedure prism.draw(xc, yc, t, phi : real; axis : boolean);
    var x, y: real;
      dy : integer;
    begin
      x := xc;
      y := yc - a / sqrt(3);
      dy := round(h * sin(phi));  {z -> y'}
      if axis then begin
         Image.Canvas.Pen.color := clblack;
         Image.Canvas.MoveTo(round(xc), round(2.3*Image.height * cos(1.22) + 30));
         Image.Canvas.Pen.Style := psDashDot;
         Image.Canvas.Pen.width := 2;
         Image.Canvas.LineTo(round(xc), round(2.3*Image.height * cos(1.22) - dy - 30));
         Image.Canvas.Pen.Style := psSolid;
         Image.Canvas.Pen.width := 4;
      end;

      Image.Canvas.Pen.color := clblue;
      Image.Canvas.MoveTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi));
      Image.Canvas.LineTo(new_x(x + a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x + a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));
      Image.Canvas.MoveTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi) - dy);
      Image.Canvas.LineTo(new_x(x + a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x + a/2, y + sqrt(3) * a / 2, xc, yc, t, phi) - dy);


      Image.Canvas.Pen.color := clgreen;
      Image.Canvas.LineTo(new_x(x - a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x - a/2, y + sqrt(3) * a / 2, xc, yc, t, phi) - dy);
      Image.Canvas.MoveTo(new_x(x + a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x + a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));
      Image.Canvas.LineTo(new_x(x - a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x - a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));

      Image.Canvas.Pen.color := clred;
      Image.Canvas.LineTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi));
      Image.Canvas.MoveTo(new_x(x - a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x - a/2, y + sqrt(3) * a / 2, xc, yc, t, phi) - dy);
      Image.Canvas.LineTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi) - dy);


      Image.Canvas.Pen.color := claqua;
      Image.Canvas.MoveTo(new_x(x - a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x - a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));
      Image.Canvas.LineTo(new_x(x - a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x - a/2, y + sqrt(3) * a / 2, xc, yc, t, phi) - dy);

      Image.Canvas.Pen.color := clpurple;
      Image.Canvas.MoveTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi));
      Image.Canvas.LineTo(new_x(x, y, xc, yc, t), new_y(x, y, xc, yc, t, phi) - dy);

      Image.Canvas.Pen.color := clfuchsia;
      Image.Canvas.MoveTo(new_x(x + a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x + a/2, y + sqrt(3) * a / 2, xc, yc, t, phi));
      Image.Canvas.LineTo(new_x(x + a / 2, y + sqrt(3) * a / 2, xc, yc, t), new_y(x + a/2, y + sqrt(3) * a / 2, xc, yc, t, phi) - dy);
```

Рисунок 6.1 – методы класса prism

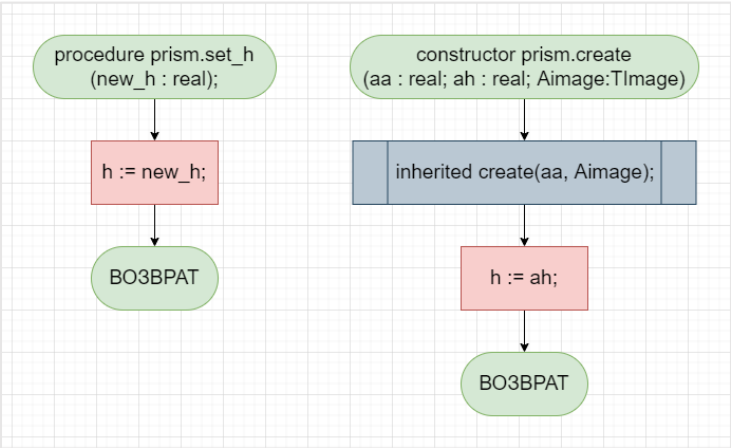

Рисунок 6.2 – схема методов prism (1)

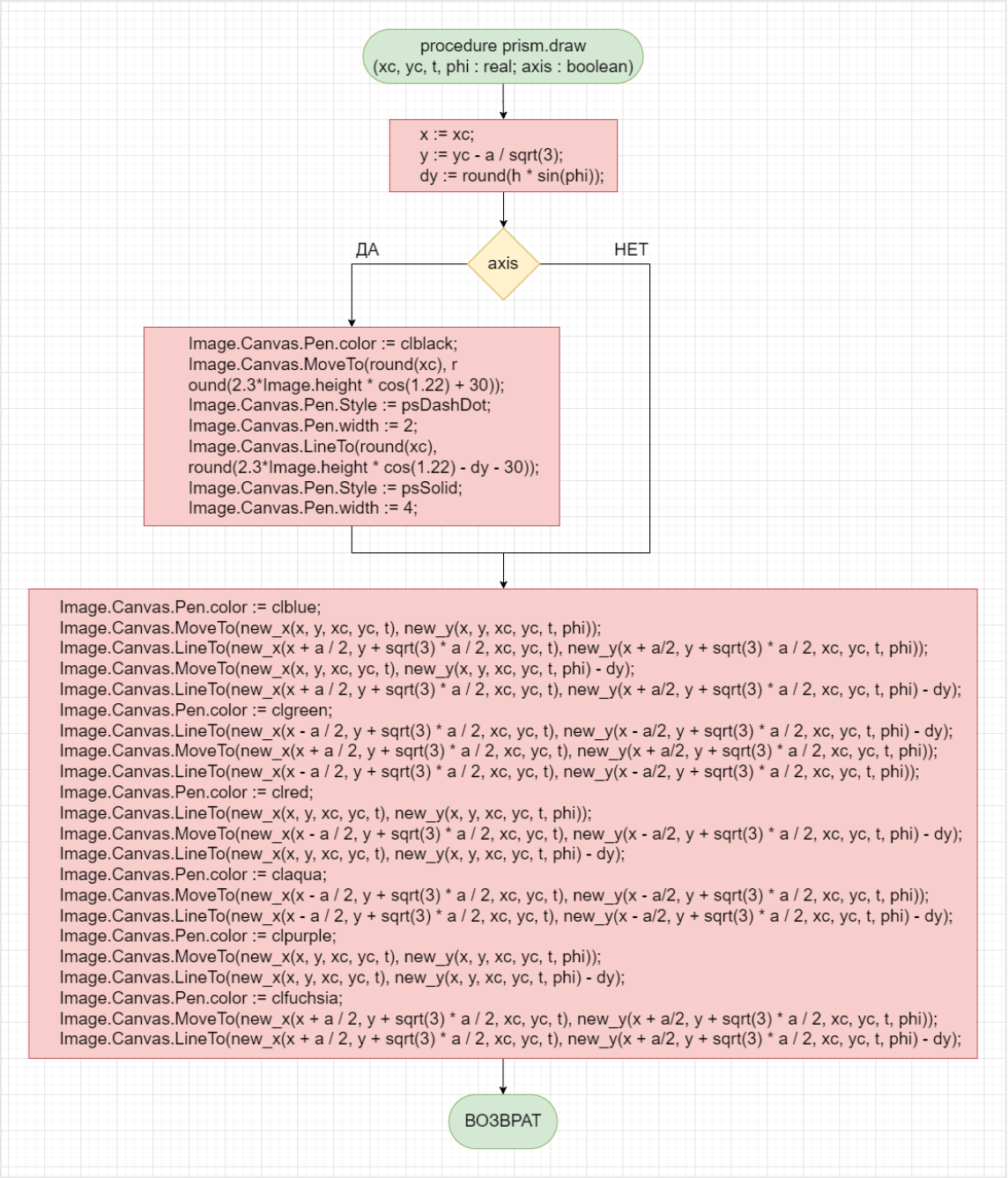

Рисунок 6.3 – схема методов prism (2)

Функции преобразования координат

```
    function new_x(x, y, x_c, y_c, t : real) : integer;
    begin
55    new_x := round((x - x_c)*cos(t) + (y - y_c)*sin(t) + x_c);
    end;

    function new_y(x, y, x_c, y_c, t, phi : real) : integer;
      begin
60    new_y := round(((y - y_c)*cos(t) - (x - x_c)*sin(t) + y_c) * cos(phi));
    end;
```

Рисунок 7.1 – код функций

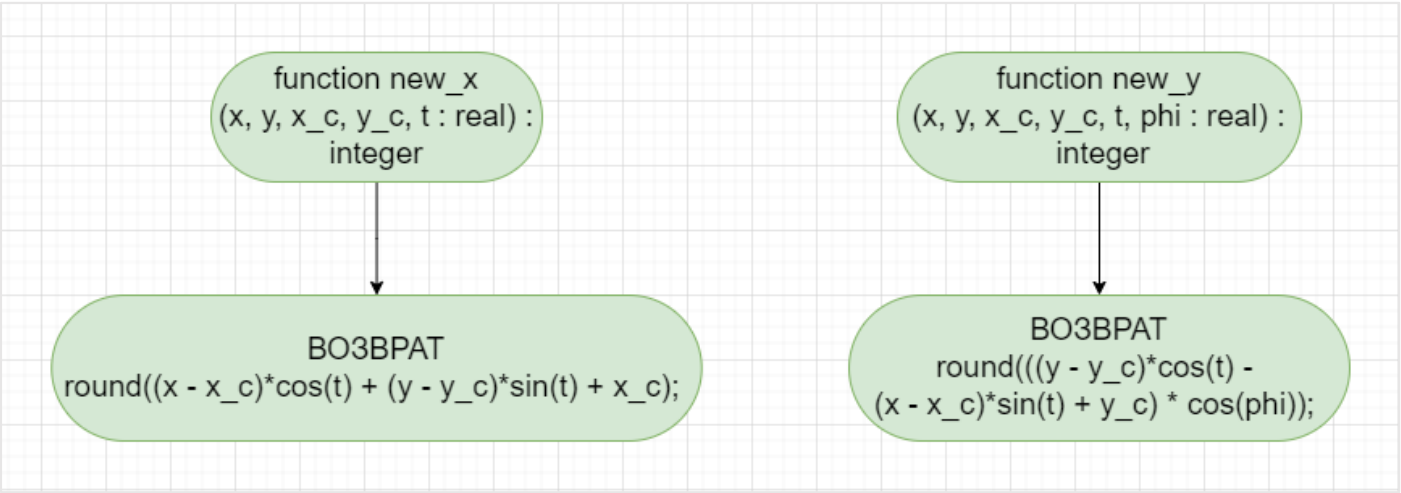

Рисунок 7.2 – схема функций

4) Добавим на форму:

- 7 текстовых полей TEdit, TLabel и 7 трекбаров TTrackbar, с помощью которых будет производиться изменение параметров фигур и вывод их текущих значений.
- Окно TImage для представления геометрических фигур.
- Таймер для обновления изображений в окне TImage.
- 3 кнопки для: старта/остановки вращения; скрытия/показа осей вращения фигур; выхода из приложения.

5) На рисунках ниже представлены граф состояния пользовательского интерфейса, объектная декомпозиция, диаграмма используемых классов VCL, код программы и интерфейс (форма) приложения.

Рисунок 8.1 – граф состояния пользовательского интерфейса

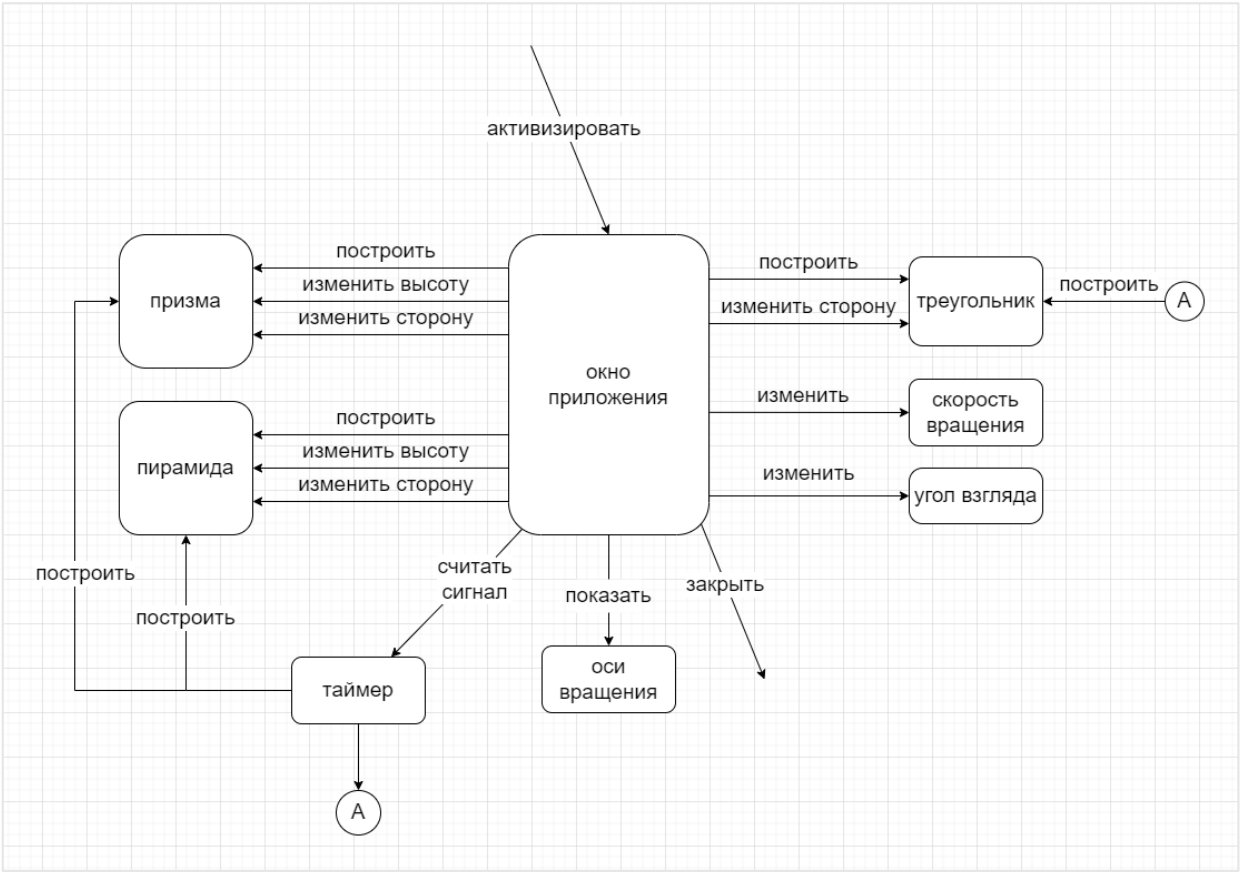

Рисунок 8.2 – объектная декомпозиция

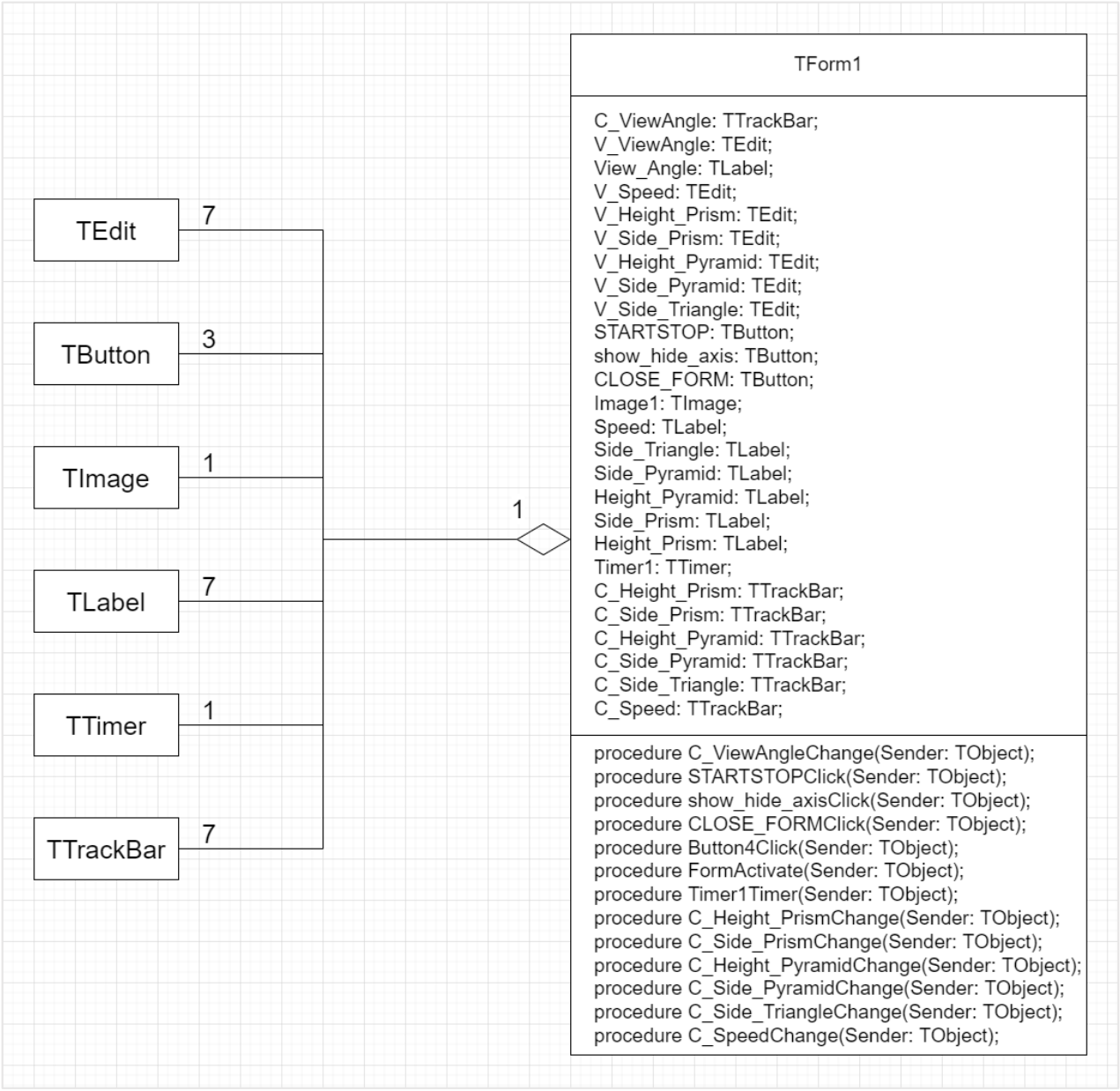

Рисунок 8.3 – диаграмма классов VCL

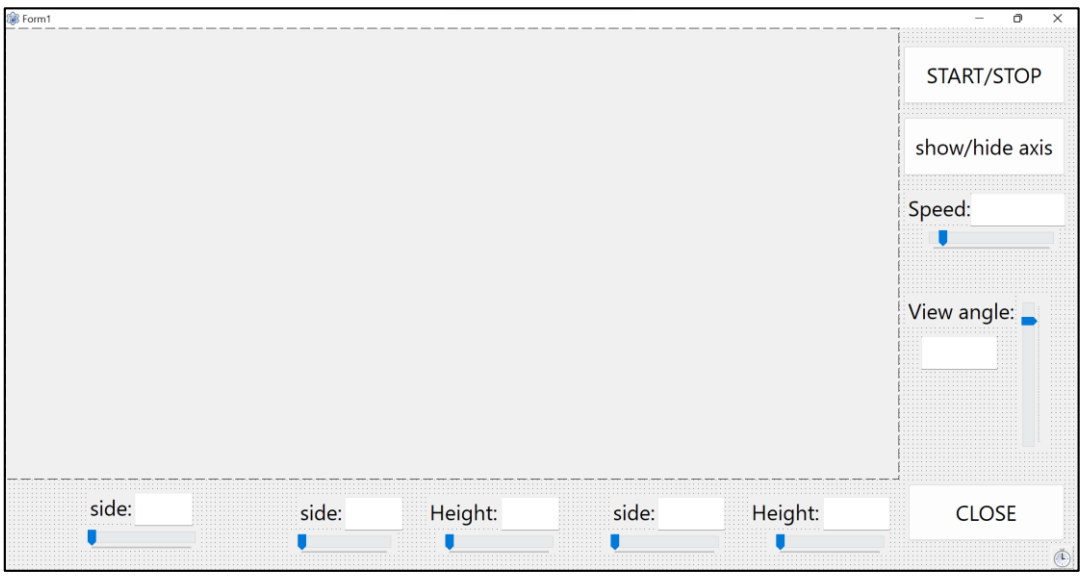

Рисунок 8.4 – форма приложения

Код программы

```
unit Unit1;

{$mode objfpc}{$H+}
interface
uses
  unit2, Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
  ExtCtrls, ComCtrls;
type

  { TForm1 }

  TForm1 = class(TForm)
    C_ViewAngle:  TTrackBar;
    V_ViewAngle:  TEdit;
    View_Angle:  TLabel;
    V_Speed: TEdit;
    V_Height_Prism: TEdit;
    V_Side_Prism: TEdit;
    V_Height_Pyramid: TEdit;
    V_Side_Pyramid: TEdit;
    V_Side_Triangle: TEdit;
    STARTSTOP: TButton;
    show_hide_axis: TButton;
    CLOSE_FORM: TButton;
    Image1: TImage;
    Speed: TLabel;
    Side_Triangle: TLabel;
    Side_Pyramid: TLabel;
    Height_Pyramid: TLabel;
    Side_Prism: TLabel;
    Height_Prism: TLabel;
    Timer1: TTimer;
    C_Height_Prism: TTrackBar;
    C_Side_Prism: TTrackBar;
    C_Height_Pyramid: TTrackBar;
    C_Side_Pyramid: TTrackBar;
    C_Side_Triangle: TTrackBar;
    C_Speed: TTrackBar;
    procedure C_ViewAngleChange(Sender: TObject);
    procedure STARTSTOPClick(Sender: TObject);
    procedure show_hide_axisClick(Sender: TObject);
    procedure CLOSE_FORMClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
```

```
    procedure Timer1Timer(Sender: TObject);
    procedure C_Height_PrismChange(Sender: TObject);
    procedure C_Side_PrismChange(Sender: TObject);
    procedure C_Height_PyramidChange(Sender: TObject);
    procedure C_Side_PyramidChange(Sender: TObject);
    procedure C_Side_TriangleChange(Sender: TObject);
    procedure C_SpeedChange(Sender: TObject);

  private
  public
  end;

var
  Form1: TForm1;
  Tr : triangle;
  Pyr : pyramid;
  Pri : prism;
  t : real = 0.0;
  dt : real = 0.05;
  phi : real = 1.22;
  s : String;
  axis : boolean = true;
implementation

{$R *.lfm}

{ TForm1 }
{My procedure}
procedure clearForm;
begin
  Form1.Image1.Canvas.Pen.color := clwindow;
  Form1.Image1.Canvas.Brush.color := clwindow;
  Form1.Image1.Canvas.Rectangle(0, 0, Form1.Image1.width, Form1.Image1.height);
  Form1.Image1.Canvas.Pen.color := clblack;
  Form1.Image1.Canvas.Brush.color := clblack;
end;


procedure TForm1.STARTSTOPClick(Sender: TObject);
begin
  Timer1.Enabled:= not Timer1.Enabled;
  Timer1.Interval := 50;
end;

procedure TForm1.C_ViewAngleChange(Sender: TObject);
begin
    phi := C_ViewAngle.Position * 3.14 / 180;
```

```
    str(phi:4:2, s);
    V_ViewAngle.text  := s + ' rad';
end;


procedure TForm1.show_hide_axisClick(Sender: TObject);
begin
  axis := not axis;
end;

procedure TForm1.CLOSE_FORMClick(Sender: TObject);
begin
  Form1.close;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin

end;

procedure TForm1.FormActivate(Sender: TObject);
begin
  clearForm();
  Tr.create(250, Image1);
  Pyr.create(300, 400, Image1);
  Pri.create(250, 400, Image1);

  C_ViewAngle.position  := round(1.22 * 180 / 3.14);
  C_Side_Triangle.position := 250;
  C_Side_Pyramid.position := 300;
  C_Side_Prism.position := 250;
  C_Height_Prism.position := 400;
  C_Height_Pyramid.position := 400;
  C_Speed.position := round(dt*1000);
  C_ViewAngle.position  := round(phi*180/3.14);

  str(dt / 0.05:4:2, s);
  V_Speed.text := s + ' rad/s';

  V_Side_Triangle.text := inttostr(250);
  V_Side_Pyramid.text := inttostr(300);
  V_Side_Prism.text := inttostr(250);
  V_Height_Prism.text := inttostr(400);
  V_Height_Pyramid.text := inttostr(400);

  str(phi:4:2, s);
  V_ViewAngle.text  := s + ' rad';
```

```
end;



procedure TForm1.Timer1Timer(Sender: TObject);
begin
  clearForm;
  Tr.draw(280, 2.3*Image1.height*cos(1.22) / cos(phi), t, phi, axis);
  Pyr.draw(Image1.width / 2, 2.3*Image1.height*cos(1.22) / cos(phi), t, phi, axis);
  Pri.draw(Image1.width - 280, 2.3*Image1.height*cos(1.22) / cos(phi), t, phi, axis);
  t := t + dt;
end;

procedure TForm1.C_Height_PrismChange(Sender: TObject);
begin
   Pri.set_h(C_Height_Prism.position);
   V_Height_Prism.text := inttostr(C_Height_Prism.position);
end;

procedure TForm1.C_Side_PrismChange(Sender: TObject);
begin
  Pri.set_a(C_Side_Prism.position);
  V_Side_Prism.text := inttostr(C_Side_Prism.position);
end;

procedure TForm1.C_Height_PyramidChange(Sender: TObject);
begin
  Pyr.set_h(C_Height_Pyramid.position);
  V_Height_Pyramid.text := inttostr(C_Height_Pyramid.position);
end;

procedure TForm1.C_Side_PyramidChange(Sender: TObject);
begin
  Pyr.set_a(C_Side_Pyramid.position);
  V_Side_Pyramid.text := inttostr(C_Side_Pyramid.position);
end;

procedure TForm1.C_Side_TriangleChange(Sender: TObject);
begin
  Tr.set_a(C_Side_Triangle.position);
  V_Side_Triangle.text := inttostr(C_Side_Triangle.position);
end;

procedure TForm1.C_SpeedChange(Sender: TObject);
begin
  dt := C_Speed.position / 1000;
  str(dt / 0.05:4:2, s);
```

```
  V_Speed.text := s + ' rad/s';
end;

end.
```

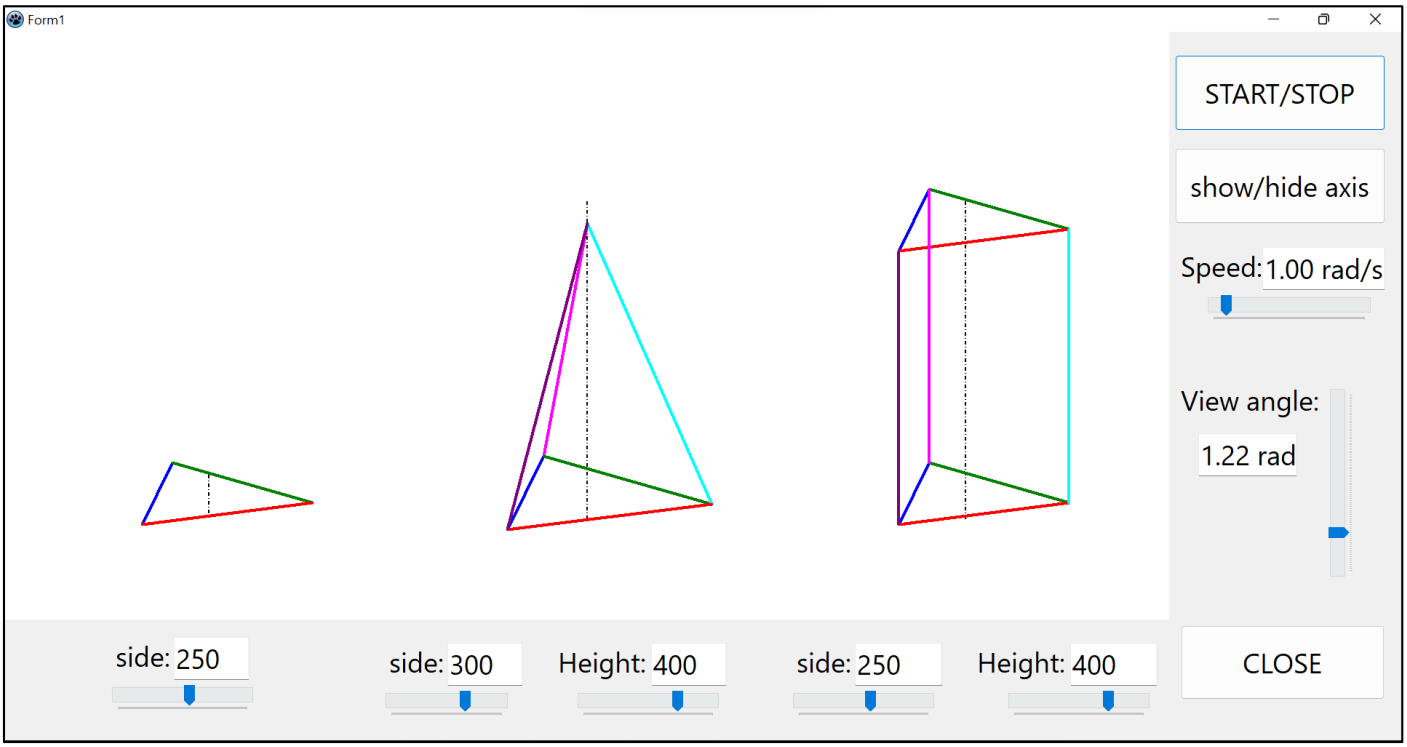

Рисунок 8.5 – пример работы приложения

Часть 1.2. Программирование с использованием метаклассов. Создание контейнеров. Использование исключений при программировании в среде Delphi

Условие задания: Моделировать стек, в качестве элементов которого могут использоваться числа и символы. Операции: добавление элемента, удаление элемента, печать элементов стека. Создать класс-потомок, который содержит процедуру сортировки элементов стека (числа по возрастанию, символы по алфавиту). Тестировать полученную модель.

В отчете представить диаграмму классов и обосновать выбранную структуру представления данных.

Решение

1) Стек основан на односвязном списке, так как в список удобно добавлять элементы и удалять их, и реализация методов для работы со списками понятная и наглядная.
2) Были реализованы три класса:

- Класс Node для хранения элемента списка
- Класс stack для хранения стека и работы с ним
- Класс NewStack, наследник класса stack, с добавленным методом сортировки

На рисунке 1.1 показана диаграмма классов.

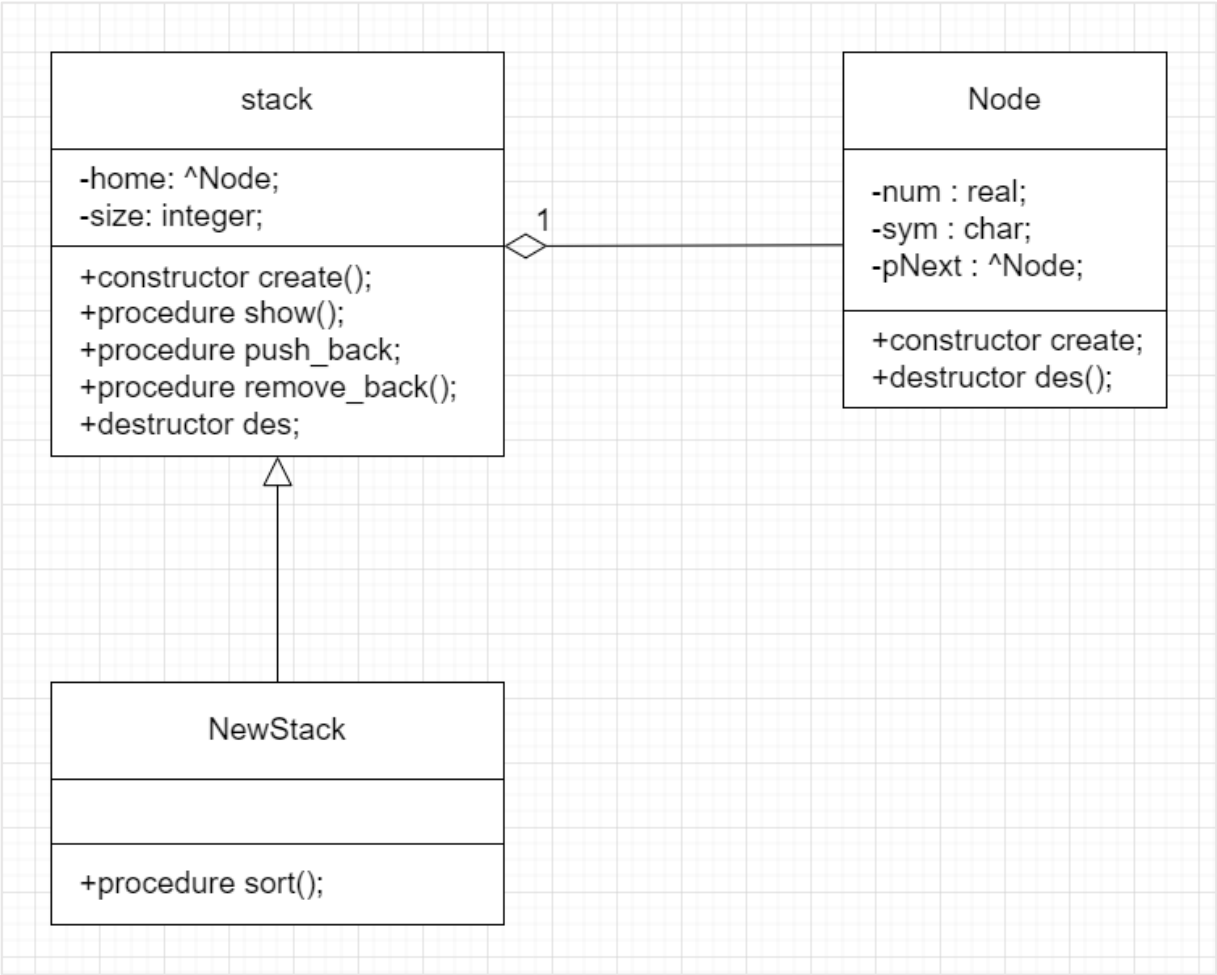

Рисунок 1.1 – диаграмма классов

3) На рисунках ниже представлены программная реализация классов, схемы методов классов, код тестирующей программы и результаты тестирования

Класс Node

```
type Node = object
  private
    num : real;
    sym : char;
    pNext : ^Node;
  public
    constructor create(Anum : real; Asym : char);
    destructor des();
  end;

destructor Node.des();
begin

end;

constructor Node.create(Anum : real; Asym : char);
begin
  num := Anum;
  sym := Asym;
  pNext := nil;
end;
```

Рисунок 2.1 – программная реализация класса Node

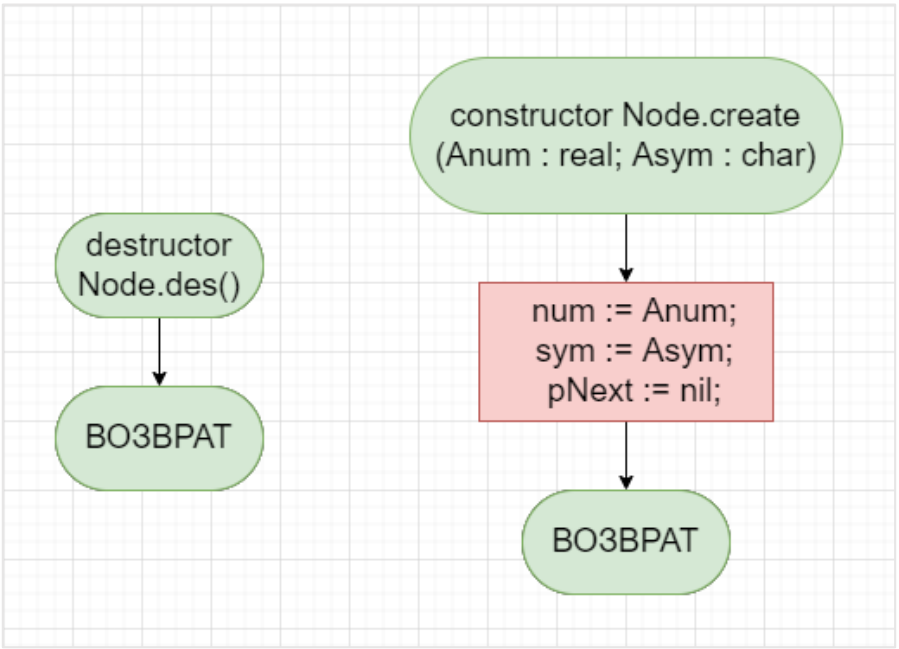

Рисунок 2.2 – схема методов класса Node

Класс stack

```
type stack = object
  private
    home: ^Node;
    size: integer;
  public
    constructor create();
    procedure show();
    procedure push_back(num : real; sym : char);
    procedure remove_back();
    destructor des;
  end;

constructor stack.create();
begin
  size := 0;
  home := nil;
end;

procedure stack.show();
var ptr : ^Node;
begin
  if home = nil then writeln('Stack is empty :(')
  else begin
    ptr := home;
    write('size: ', size, ' | ');
    while ptr <> nil do begin
      write('{', ptr^.num:7:3, '; "', ptr^.sym, '"} ');
      ptr := ptr^.pNext;
    end;
    writeln;
  end;
end;
```

```
procedure stack.push_back(num : real; sym : char);
var ptr : ^Node;
begin
  if home = nil then new(home, create(num, sym))
    else begin
      ptr := home;
      while ptr^.pNext <> nil do ptr := ptr^.pNext;
      new(ptr^.pNext, create(num, sym));
    end;
  inc(size);
end;

procedure stack.remove_back();
var ptr, del : ^Node;
begin
  if home <> nil then
  begin
    ptr := home;
    if home^.pNext = nil then self.des
    else begin
      while (ptr^.pNext)^.pNext <> nil do ptr := ptr^.pNext;
      del := ptr^.pNext;
      dispose(del, des);
      ptr^.pNext := nil;
    end;
    dec(size);
  end;
end;

destructor stack.des();
var ptr : ^Node;
begin
  while home <> nil do begin
     ptr := home;
     dispose(home, des);
     home := ptr^.pNext;
  end;
end;
```

Рисунок 3.1 – программная реализация класса stack

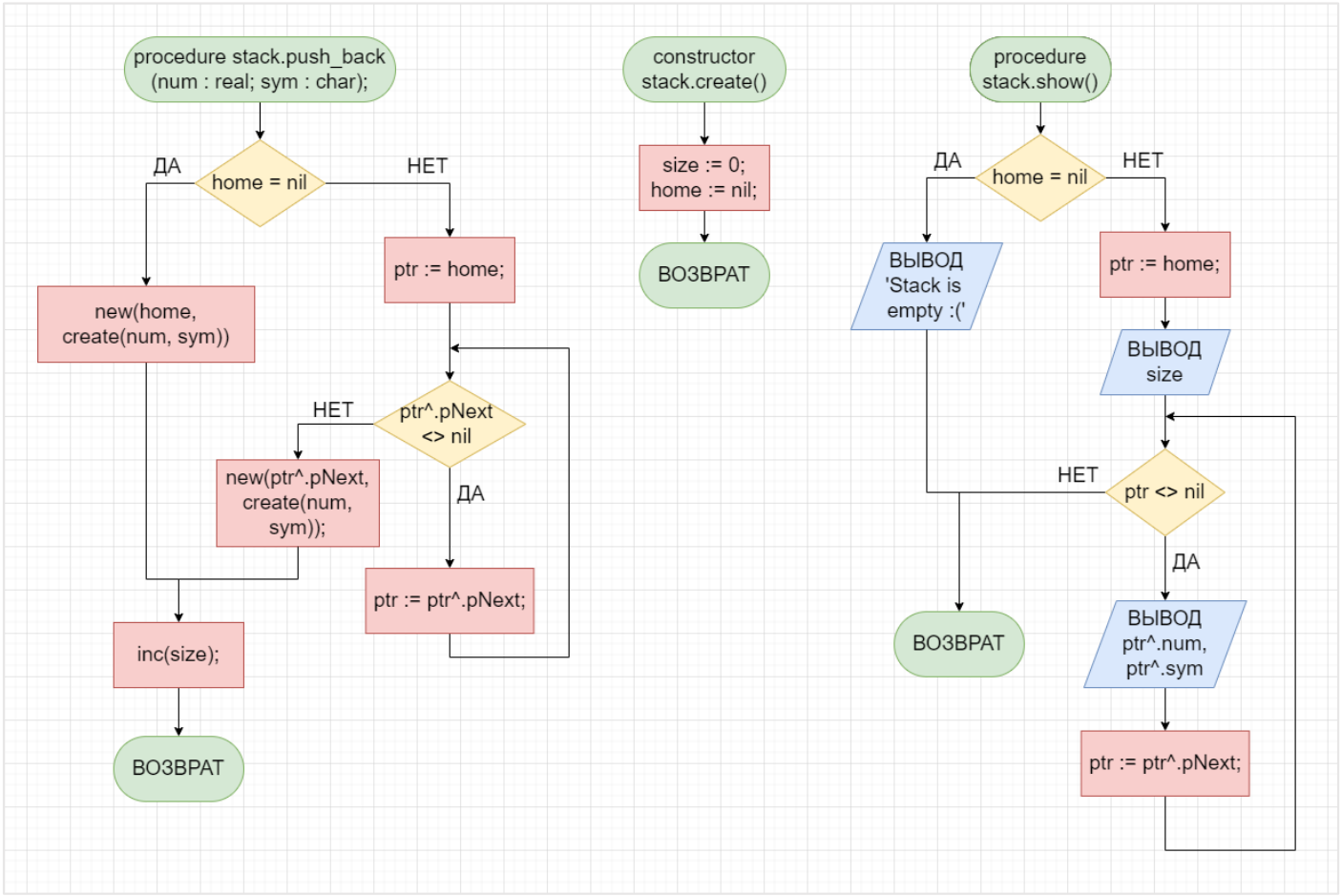

Рисунок 3.2 – схема методов класса stack (1)

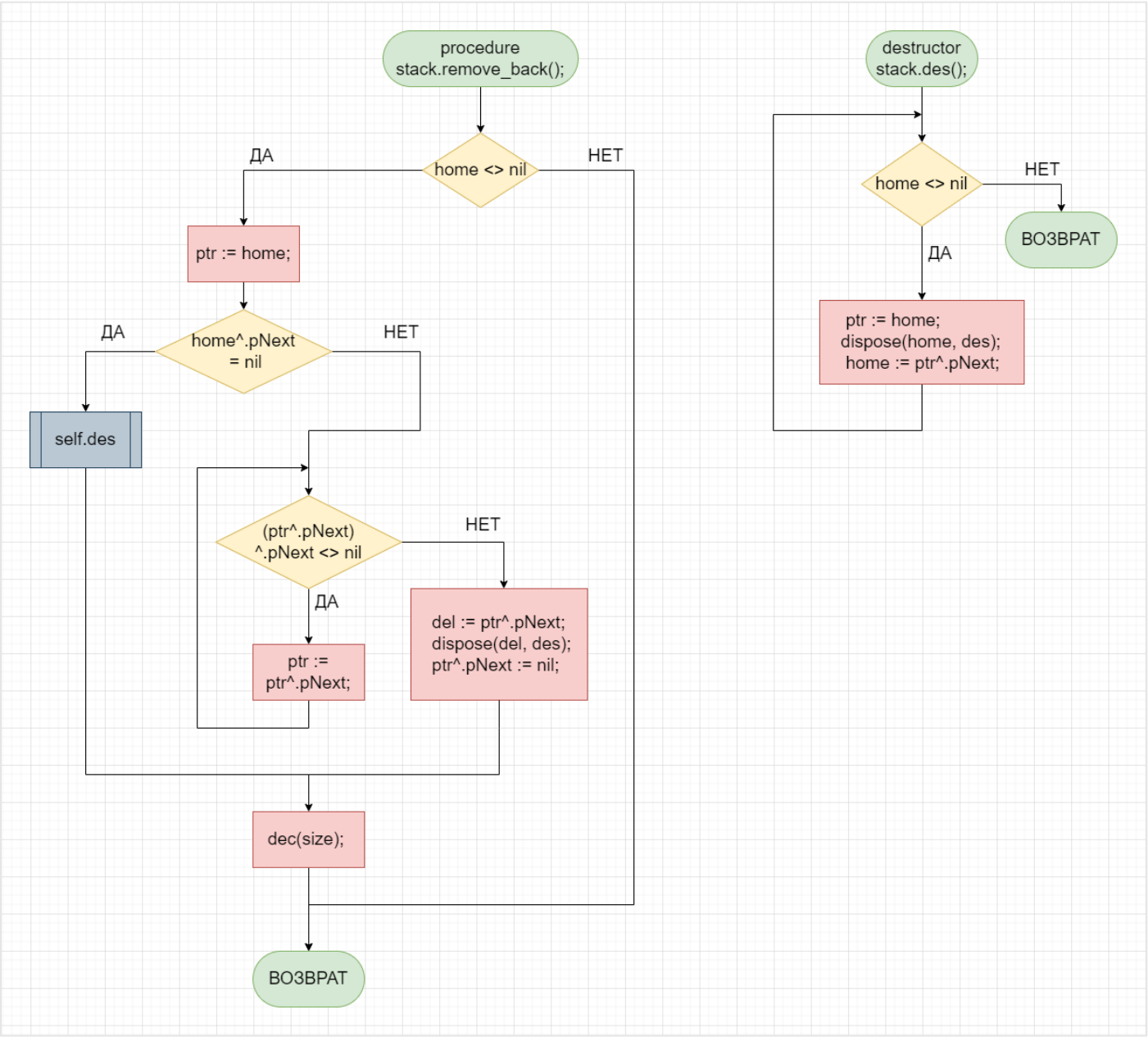

Рисунок 3.3 – схема методов класса stack (2)

Класс NewStack

```
type NewStack = object(stack)
  private
  public
     procedure sort();
  end;


procedure NewStack.sort();
var i, j : integer;
    j_1_ptr, j_ptr : ^Node;
    num : real;
    sym : char;
begin
    for i := 1 to size - 1 do begin
       j_1_ptr := home;
       j_ptr := j_1_ptr^.pNext;
       for j := 1 to size - i do begin
         if (j_ptr^.num < j_1_ptr^.num) or ((j_ptr^.num = j_1_ptr^.num) and (j_ptr^.sym < j_1_ptr^.sym)) then begin
            num := j_ptr^.num;
            sym := j_ptr^.sym;
            j_ptr^.num := j_1_ptr^.num;
            j_ptr^.sym := j_1_ptr^.sym;
            j_1_ptr^.num := num;
            j_1_ptr^.sym := sym;
         end;
         j_1_ptr := j_1_ptr^.pNext;
         j_ptr := j_ptr^.pNext;
       end;
    end;
end;
```

Рисунок 4.1 – программная реализация класса NewStack

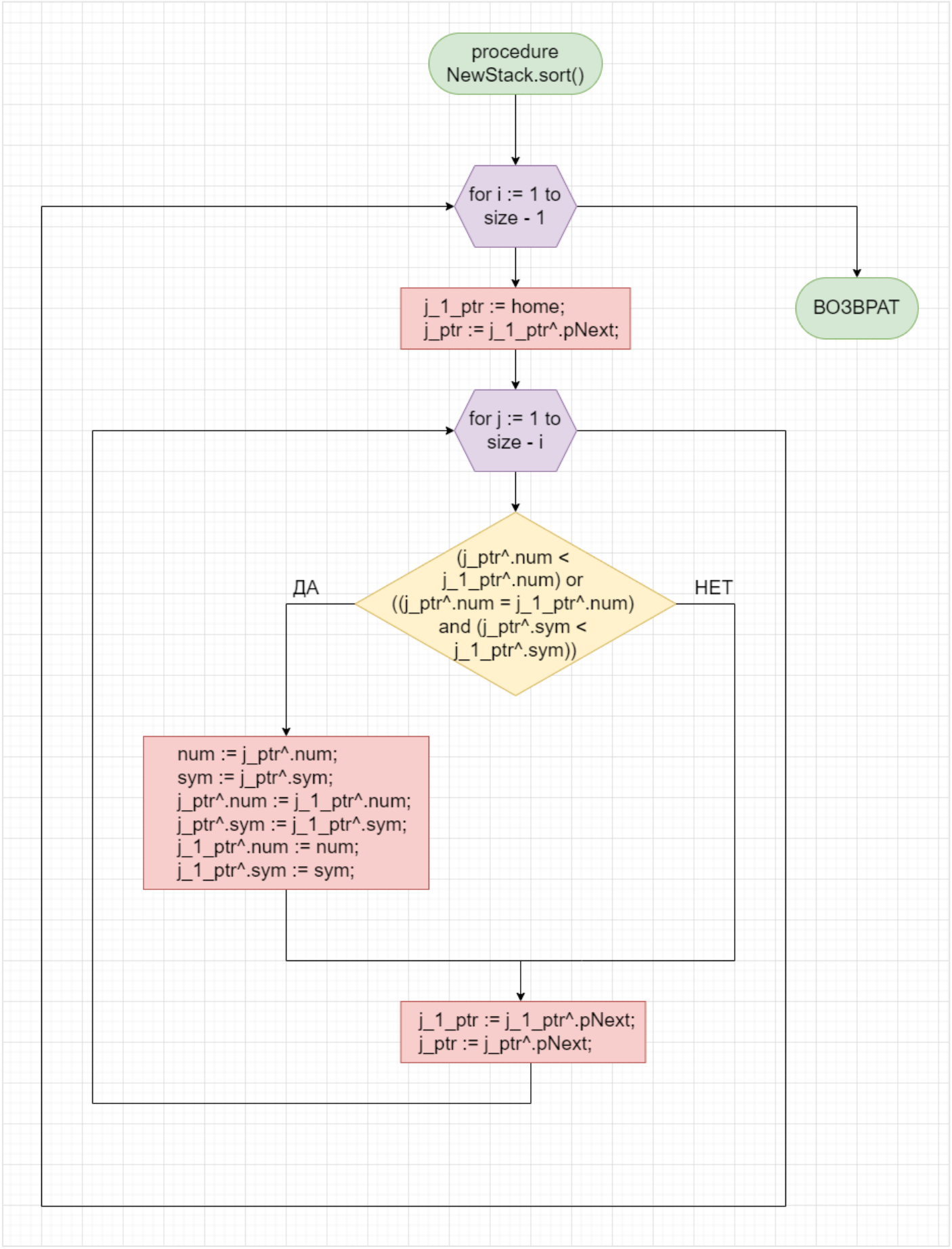

Рисунок 4.2 – схема методов класса NewStack

Тестирующая программа и результаты тестирования

```
    var st : NewStack;
130   i, n : integer;
      sym : char;
      num : real;
    begin
      randomize;
135   write('Enter size of stack: ');
      readln(n);
      for i := 1 to n do begin
          num := real((random(20000) - 10000) / 100);
          sym := char(random(68) + 60);
140       st.push_back(num, sym);
      end;
      write('Input stack: ');
      st.show;
      write('Stack with deleted element: ');
145   st.remove_back;
      st.show;
      write('Sort stack: ');
      st.sort;
      st.show;
150   st.des;
      readln;
    end.
```

Рисунок 5.1 – код тестирующей программы

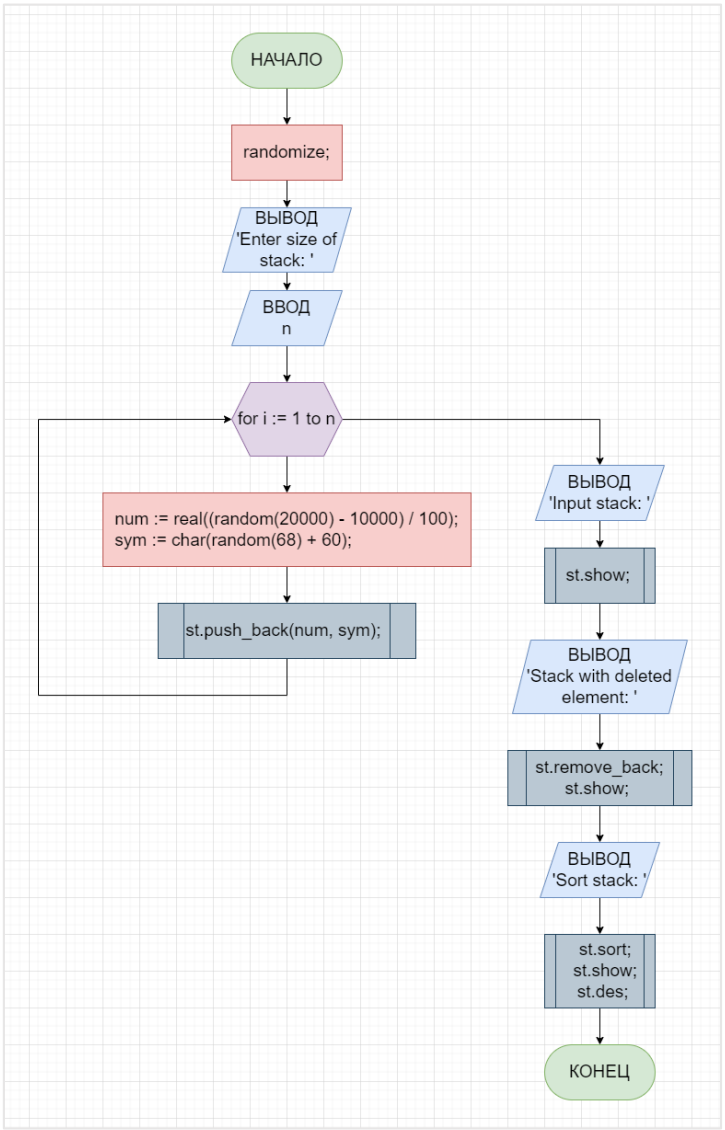

Рисунок 5.2 – схема алгоритма

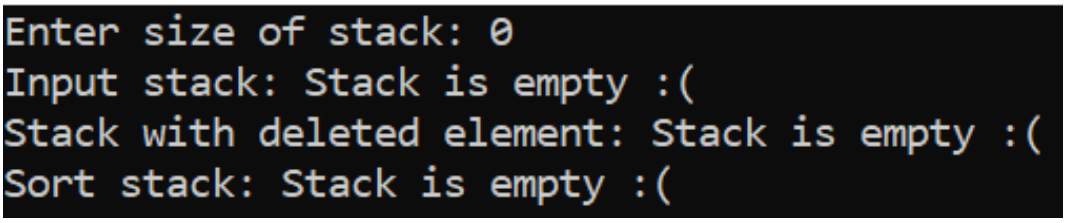

Рисунок 5.3 – результат тестирования (size = 0)

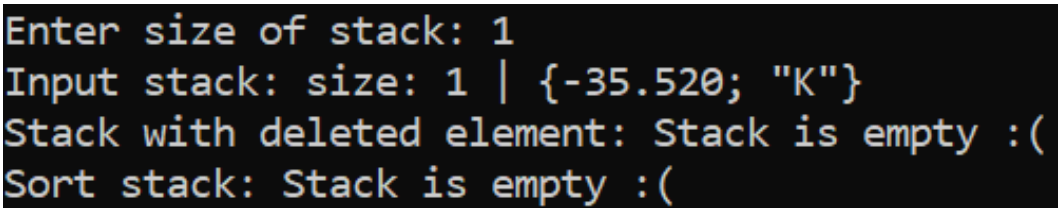

Рисунок 5.4 – результат тестирования (size = 1)

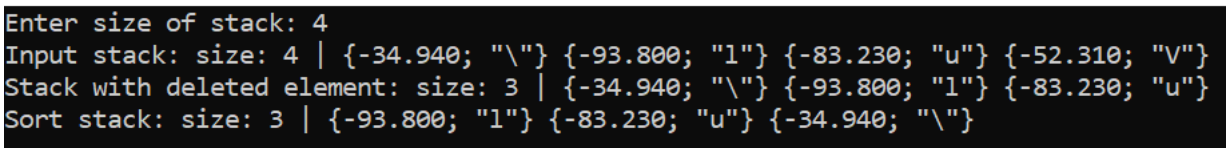

Рисунок 5.5 – результат тестирования (size = 4)

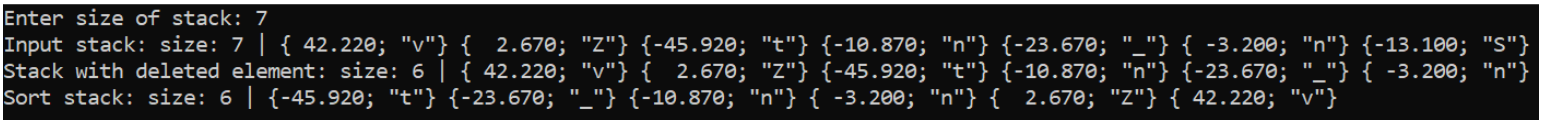

Рисунок 5.6 – результат тестирования (size = 7)

Результаты тестирований показали корректность работы программы.