*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)\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по домашней работе № 1**

**Название домашней работы: *Графика. Обработка события нажатия клавиши мыши. Наследование. Полиморфизм***

**Дисциплина: Объектно-ориентированное программирование**

Студент гр. ИУ6-22Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Плютто**



(Подпись, дата) (И.О. Фамилия)



Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. А. Веселовская**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Задание 1**

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

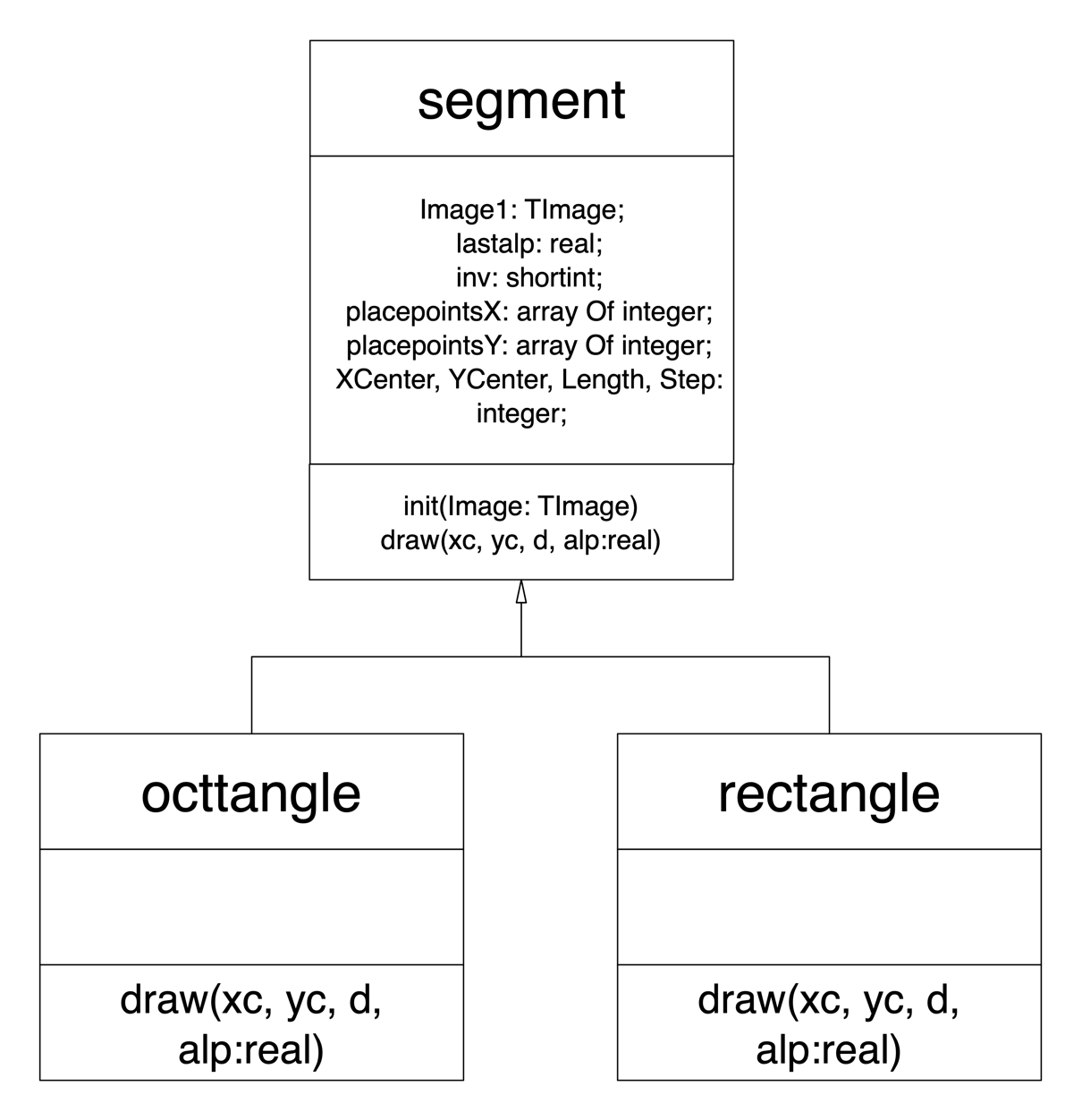
отрезок, квадрат, правильный восьмиугольник.

Реализуя механизм полиморфизма, привести объекты в одновременное вращение вокруг их геометрических центров с различными угловыми скоростями. Скорость вращения регулируется с использованием интерфейсных элементов.

В отчете привести диаграмму используемых классов Delphi и разработанных классов, граф состояний пользовательского интерфейса и объектную декомпозицию.

Решение:

Решение я решил реализовать с помощью 3 TImage, по одному TImage для каждой фигуры. Фигуры описал в отдельных классах. Вот их объявление:



Type segment = Object

Image1: TImage;

lastalp: real;

inv: shortint;

placepointsX: array [0..8] Of integer;

placepointsY: array [0..8] Of integer;

XCenter, YCenter, Length, Step: integer;

*constructor* init(Image: TImage);

*Procedure* draw(xc, yc, d, alp:real);

End;

Type octtangle = Object(segment)

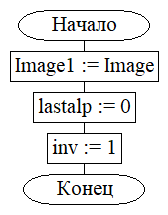
*Procedure* draw(xc, yc, d, alp:real);

End;

Type rectangle = Object(segment)

*Procedure* draw(xc, yc, d, alp:real);

End;



Каждая переменная в классе отвечает за текущее положение фигуры.

* Image1 это полотно фигуры, там, где она будет отрисовываться
* Lastalp последний угол фигуры
* Inv инверирование движения фигуры
* PlacepointsX, PlacepointsY массивы, содержащие координаты точек фигур
* XCenter, YCenter положение фигуры по центру
* Length длина стороны/отрезка
* Step шаг изменения угла за один период

Я добавил процедуру отрисовки фигур. Работает она так: на вход подаются 4 параметра: координаты центра фигуры по X и Y, длина стороны (общая длина для отрезка) и угол в радианах на который эта фигура должна быть повернута. Сама процедура переопределяет текущий угол фигуры lastalp, задает точки фигуры и отрисовывает ее.

*Procedure* segment.draw(xc, yc, d, alp:real);

Begin

self.lastalp := alp;

alp := -alp\*inv;

self.placepointsX[0] := round(xc+d\*cos(alp));

self.placepointsX[1] := round(xc-d\*cos(alp));

self.placepointsY[0] := round(yc-d\*sin(alp));

self.placepointsY[1] := round(yc+d\*sin(alp));

Image1.canvas.MoveTo(self.placepointsX[0], self.placepointsY[0]);

Image1.canvas.LineTo(self.placepointsX[1], self.placepointsY[1]);

End;

*Procedure* rectangle.draw(xc, yc, d, alp:real);

Var x1, x2, y1, y2, x, y: real;

i: integer;

Begin

self.lastalp := alp;

alp := alp\*inv;

x1 := xc-d/2;

x2 := xc+d/2;

y1 := yc-d/2;

y2 := yc+d/2;

*For* i:=0 To 4 *Do*

Begin

Case i Of

0:

Begin

x := x1;

y := y1;

End;

1:

Begin

x := x2;

y := y1;

End;

2:

Begin

x := x2;

y := y2;

End;

3:

Begin

x := x1;

y := y2;

End;

4:

Begin

x := x1;

y := y1;

End;

End;

self.placepointsX[i] := round(xc + (x - xc) \* cos(alp) - (y - yc) \* sin(alp));

self.placepointsY[i] := round(yc + (x - xc) \* sin(alp) + (y - yc) \* cos(alp));

If i=0 Then Image1.canvas.MoveTo(self.placepointsX[i], self.placepointsY[i])

Else Image1.canvas.LineTo(self.placepointsX[i], self.placepointsY[i]);

End;

End;

*Procedure* octtangle.draw(xc, yc, d, alp:real);

Var x1, x2, x3, x4, y1, y2, y3, y4, x, y: real;

i: integer;

Begin

self.lastalp := alp;

alp := alp\*inv;

x1 := xc-d\*sin(pi/4);

x2 := xc-d;

x3 := xc+d\*sin(pi/4);

x4 := xc+d;

y1 := yc+d\*sin(pi/4);

y2 := yc-d\*sin(pi/4);

y3 := yc-d;

y4 := yc+d;

*For* i:=0 To 8 *Do*

Begin

Case i Of

0:

Begin

x := x1;

y := y1;

End;

1:

Begin

x := x2;

y := yc;

End;

2:

Begin

x := x1;

y := y2;

End;

3:

Begin

x := xc;

y := y3;

End;

4:

Begin

x := x3;

y := y2;

End;

5:

Begin

x := x4;

y := yc;

End;

6:

Begin

x := x3;

y := y1;

End;

7:

Begin

x := xc;

y := y4;

End;

8:

Begin

x := x1;

y := y1;

End;

End;

self.placepointsX[i] := round(xc + (x - xc) \* cos(alp) - (y - yc) \* sin(

alp));

self.placepointsY[i] := round(yc + (x - xc) \* sin(alp) + (y - yc) \* cos(

alp));

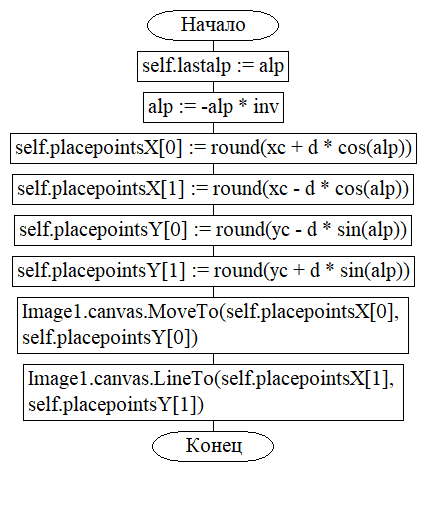
If i=0 Then Image1.canvas.MoveTo(self.placepointsX[i], self.placepointsY[

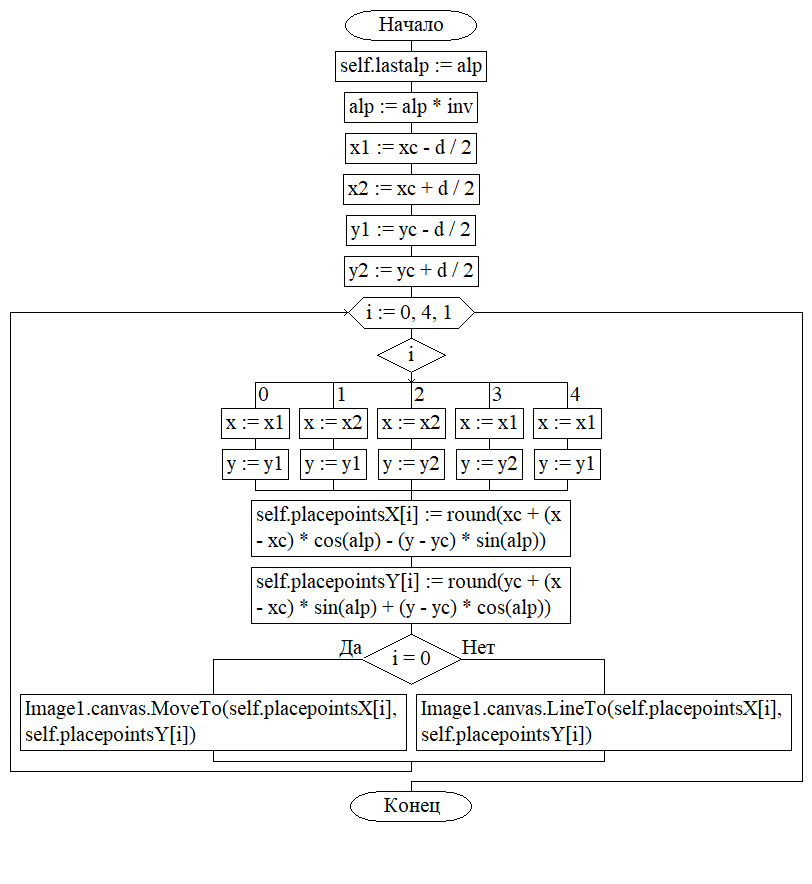
i])

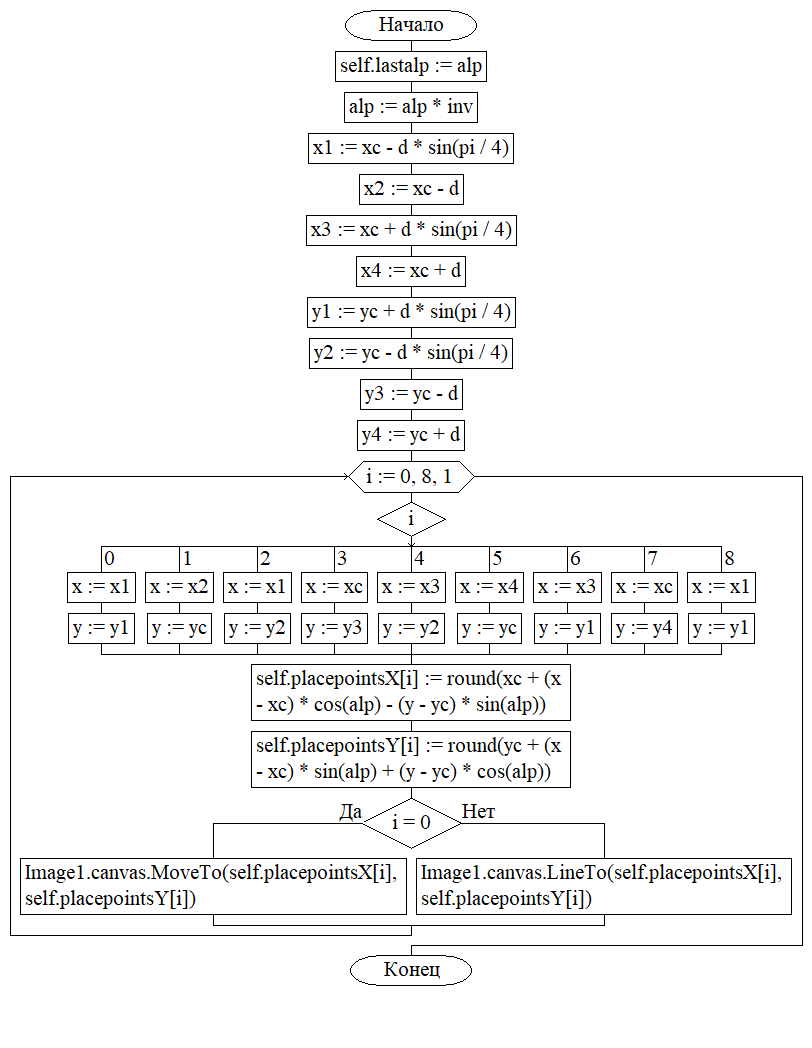
Else Image1.canvas.LineTo(self.placepointsX[i], self.placepointsY[i]);

End;

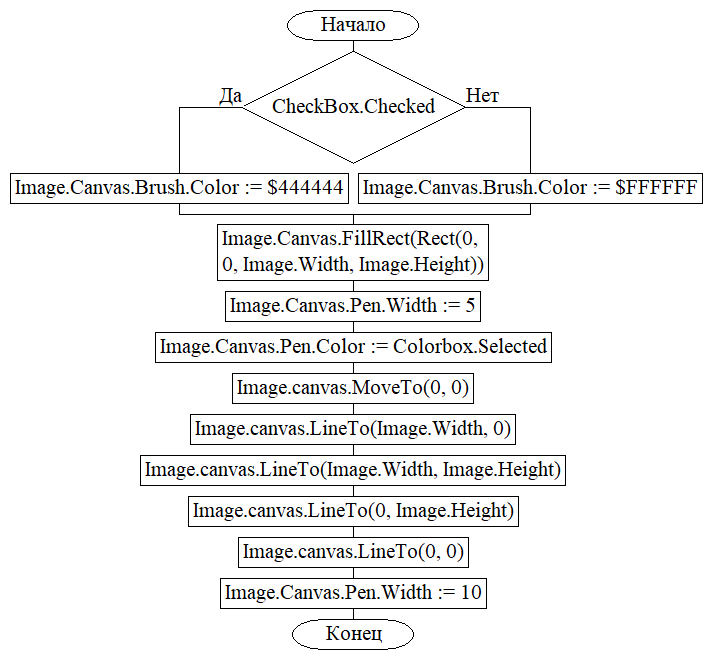
End;







Чтобы фигура двигалась необходимо с каким-то периодом изменять ее угол на какой-то шаг. При этом предыдущее положение фигуры должно быть стерто. Для этого я создал процедуру clear.



*Procedure* clear(image:TImage; CheckBox:TCheckBox; ColorBox:TColorBox);

Begin

If CheckBox.Checked Then Image.Canvas.Brush.Color := $444444

Else Image.Canvas.Brush.Color := $FFFFFF;

Image.Canvas.FillRect(Rect(0, 0, Image.Width, Image.Height));

Image.Canvas.Pen.Width := 5;

Image.Canvas.Pen.Color := Colorbox.Selected;

Image.canvas.MoveTo(0, 0) ;

Image.canvas.LineTo(Image.Width, 0);

Image.canvas.LineTo(Image.Width, Image.Height);

Image.canvas.LineTo(0, Image.Height);

Image.canvas.LineTo(0, 0);

Image.Canvas.Pen.Width := 10;

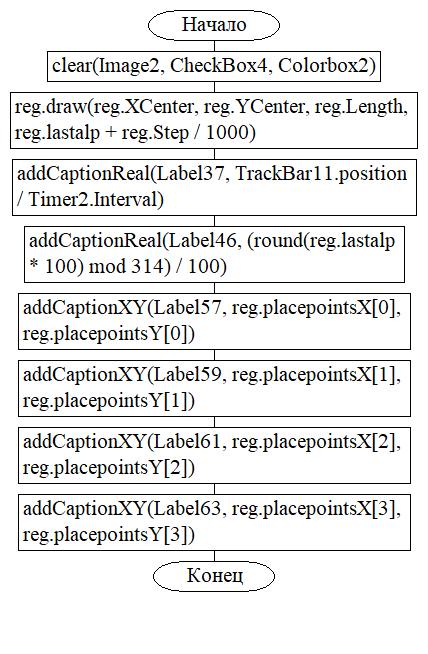
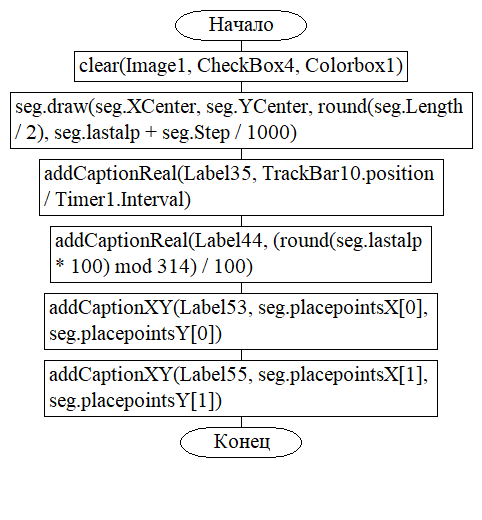
End;

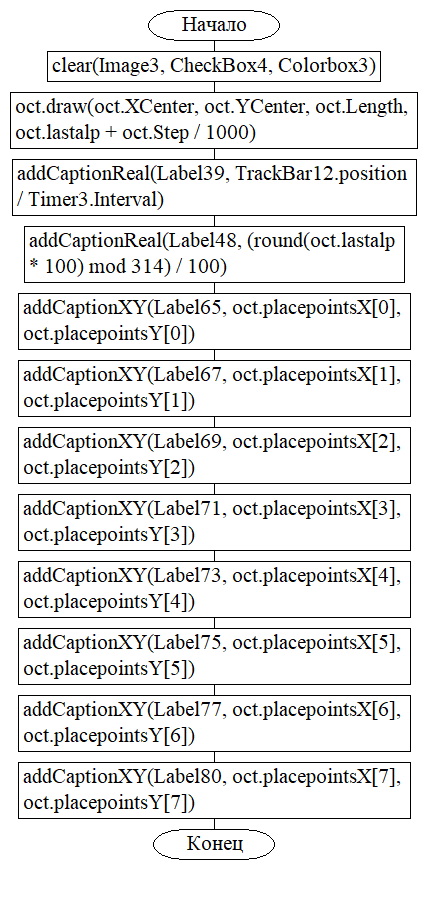
Она задает цвет фона в зависимости от выбранной темы, затем заливает этим цветом все полотно, отрисовывает рамку полотна, и задет цвет и размер кист

Далее для постоянной отрисовки я создал 3 таймера TTimer. Управлять их периодом можно так: Timer1.Interval := TrackBar13.position;

Шаг я реализовал с помощью переменной Step в каждой из фигур. По прошествии периода выполняется скрипт, который вызывает функцию очистки, отрисовки фигуры и вывода на экран некоторых параметров фигуры:

* Скорость
* Текущий угол
* Координаты всех точек





*Procedure* TForm1.Timer1Timer(Sender: TObject);

Begin

clear(Image1, CheckBox4, Colorbox1);

seg.draw(seg.XCenter,seg.YCenter,round(seg.Length/2),seg.lastalp+seg.Step/1000

);

addCaptionReal(Label35, TrackBar10.position/Timer1.Interval);

addCaptionReal(Label44, (round(seg.lastalp\*100) mod 314)/100);

addCaptionXY(Label53, seg.placepointsX[0], seg.placepointsY[0]);

addCaptionXY(Label55, seg.placepointsX[1], seg.placepointsY[1]);

End;

*Procedure* TForm1.Timer2Timer(Sender: TObject);

Begin

clear(Image2, CheckBox4, Colorbox2);

reg.draw(reg.XCenter,reg.YCenter,reg.Length,reg.lastalp+reg.Step/1000);

addCaptionReal(Label37, TrackBar11.position/Timer2.Interval);

addCaptionReal(Label46, (round(reg.lastalp\*100) mod 314)/100);

addCaptionXY(Label57, reg.placepointsX[0], reg.placepointsY[0]);

addCaptionXY(Label59, reg.placepointsX[1], reg.placepointsY[1]);

addCaptionXY(Label61, reg.placepointsX[2], reg.placepointsY[2]);

addCaptionXY(Label63, reg.placepointsX[3], reg.placepointsY[3]);

End;

*Procedure* TForm1.Timer3Timer(Sender: TObject);

Begin

clear(Image3, CheckBox4, Colorbox3);

oct.draw(oct.XCenter,oct.YCenter,oct.Length,oct.lastalp+oct.Step/1000);

addCaptionReal(Label39, TrackBar12.position/Timer3.Interval);

addCaptionReal(Label48, (round(oct.lastalp\*100) mod 314)/100);

addCaptionXY(Label65, oct.placepointsX[0], oct.placepointsY[0]);

addCaptionXY(Label67, oct.placepointsX[1], oct.placepointsY[1]);

addCaptionXY(Label69, oct.placepointsX[2], oct.placepointsY[2]);

addCaptionXY(Label71, oct.placepointsX[3], oct.placepointsY[3]);

addCaptionXY(Label73, oct.placepointsX[4], oct.placepointsY[4]);

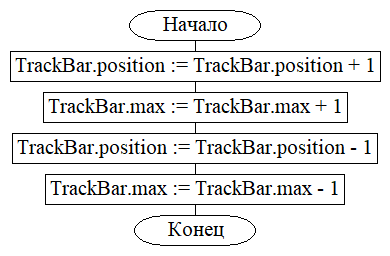
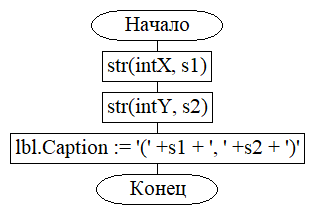
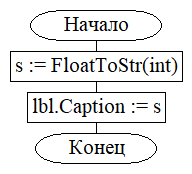
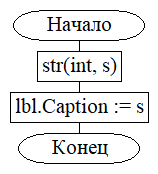
addCaptionXY(Label75, oct.placepointsX[5], oct.placepointsY[5]);

addCaptionXY(Label77, oct.placepointsX[6], oct.placepointsY[6]);

addCaptionXY(Label80, oct.placepointsX[7], oct.placepointsY[7]);

End;

Так же я реализовал функции простого вывода целых и вещественных чисел и координат точек в Label. Перезагрузку ползунков (для изменения цвета их фона или для изменения их длины)



*Procedure* addCaptionReal(lbl:TLabel; int:real);

Var s: string[10];

Begin

s := FloatToStr(int);

lbl.Caption := s

End;

*Procedure* addCaption(lbl:TLabel; int:integer);

Var s: string[4];

Begin

str(int,s);

lbl.Caption := s

End;

*Procedure* addCaptionXY(lbl:TLabel; intX, intY:integer);

Var s1,s2: string[4];

Begin

str(intX,s1);

str(intY,s2);

lbl.Caption := '('+s1+', '+s2+')'

End;

*Procedure* reloadBar(TrackBar:TTrackBar);

Begin

TrackBar.position := TrackBar.position+1;

TrackBar.max := TrackBar.max+1;

TrackBar.position := TrackBar.position-1;

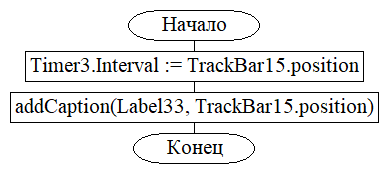
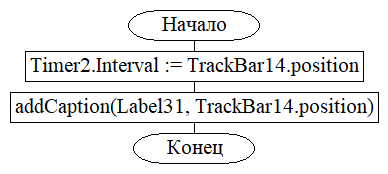
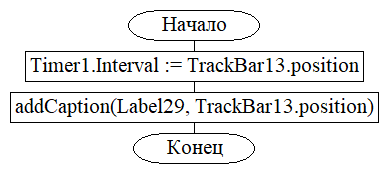
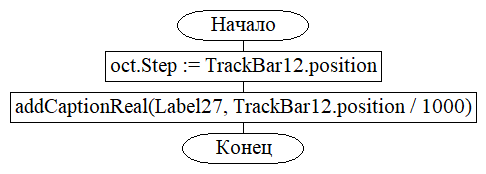
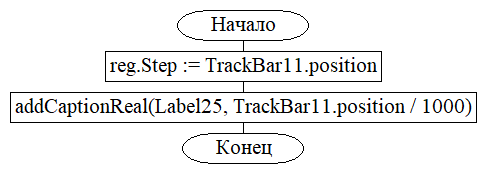
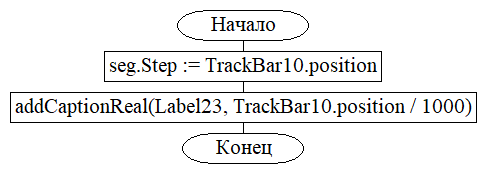
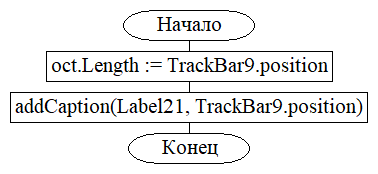
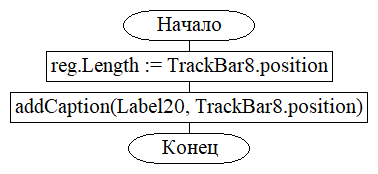
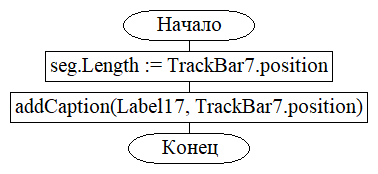
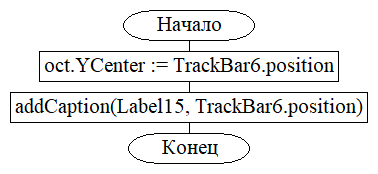
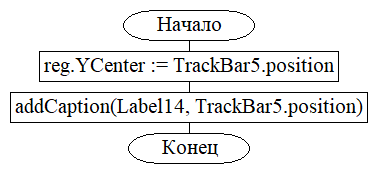
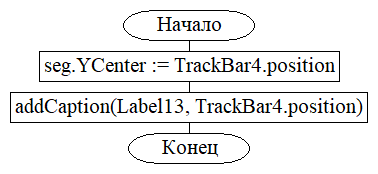
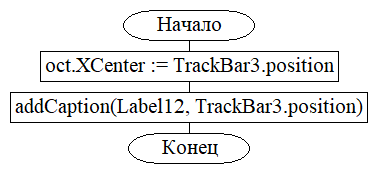
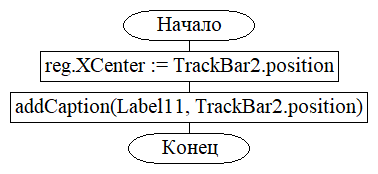
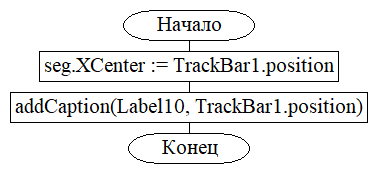
TrackBar.max := TrackBar.max-1;

End;

Ползунки отвечают за изменения параметров фигур, отдельно для каждой фигуры можно изменять:

* Координаты центра
* Длину стороны/отрезка
* Шаг вращения
* Период вращения

Пояснение: шаг и период вращения отвечают за скорость, я не вынес скорость отдельным ползунком потому что одной и той же скорости можно добиться при шаге n рад и периоде t мс, так и при шаге kn рад и периоде kn секунд, но при k>1 нагрузка на систему возрастет, а фигура станет двигаться плавнее, а при k<1 нагрузка упадет, но фигура станет двигаться более дергано.



*Procedure* TForm1.TrackBar1Change(Sender: TObject);

Begin

seg.XCenter := TrackBar1.position;

addCaption(Label10, TrackBar1.position);

End;

*Procedure* TForm1.TrackBar2Change(Sender: TObject);

Begin

reg.XCenter := TrackBar2.position;

addCaption(Label11, TrackBar2.position);

End;

*Procedure* TForm1.TrackBar3Change(Sender: TObject);

Begin

oct.XCenter := TrackBar3.position;

addCaption(Label12, TrackBar3.position);

End;

*Procedure* TForm1.TrackBar4Change(Sender: TObject);

Begin

seg.YCenter := TrackBar4.position;

addCaption(Label13, TrackBar4.position);

End;

*Procedure* TForm1.TrackBar5Change(Sender: TObject);

Begin

reg.YCenter := TrackBar5.position;

addCaption(Label14, TrackBar5.position);

End;

*Procedure* TForm1.TrackBar6Change(Sender: TObject);

Begin

oct.YCenter := TrackBar6.position;

addCaption(Label15, TrackBar6.position);

End;

*Procedure* TForm1.TrackBar7Change(Sender: TObject);

Begin

seg.Length := TrackBar7.position;

addCaption(Label17, TrackBar7.position);

End;

*Procedure* TForm1.TrackBar8Change(Sender: TObject);

Begin

reg.Length := TrackBar8.position;

addCaption(Label20, TrackBar8.position);

End;

*Procedure* TForm1.TrackBar9Change(Sender: TObject);

Begin

oct.Length := TrackBar9.position;

addCaption(Label21, TrackBar9.position);

End;

*Procedure* TForm1.TrackBar10Change(Sender: TObject);

Begin

seg.Step := TrackBar10.position;

addCaptionReal(Label23, TrackBar10.position/1000);

End;

*Procedure* TForm1.TrackBar11Change(Sender: TObject);

Begin

reg.Step := TrackBar11.position;

addCaptionReal(Label25, TrackBar11.position/1000);

End;

*Procedure* TForm1.TrackBar12Change(Sender: TObject);

Begin

oct.Step := TrackBar12.position;

addCaptionReal(Label27, TrackBar12.position/1000);

End;

*Procedure* TForm1.TrackBar13Change(Sender: TObject);

Begin

Timer1.Interval := TrackBar13.position;

addCaption(Label29, TrackBar13.position);

End;

*Procedure* TForm1.TrackBar14Change(Sender: TObject);

Begin

Timer2.Interval := TrackBar14.position;

addCaption(Label31, TrackBar14.position);

End;

*Procedure* TForm1.TrackBar15Change(Sender: TObject);

Begin

Timer3.Interval := TrackBar15.position;

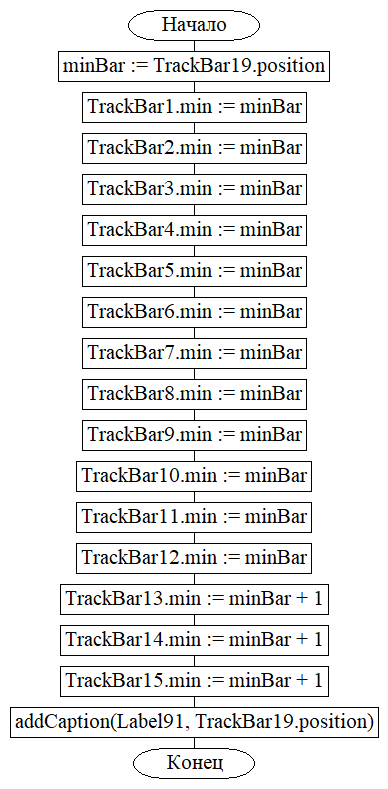
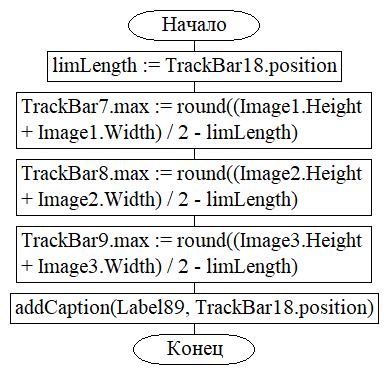
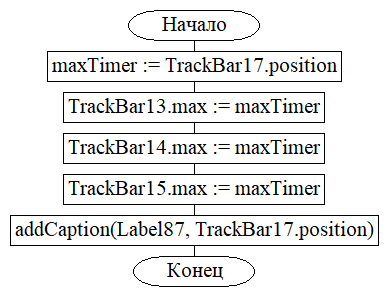
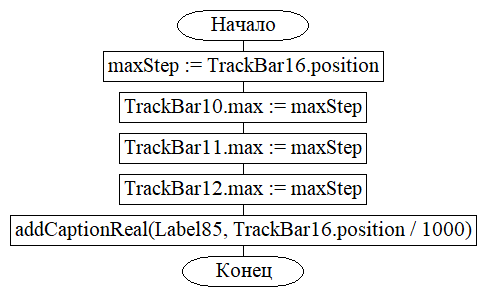
addCaption(Label33, TrackBar15.position);

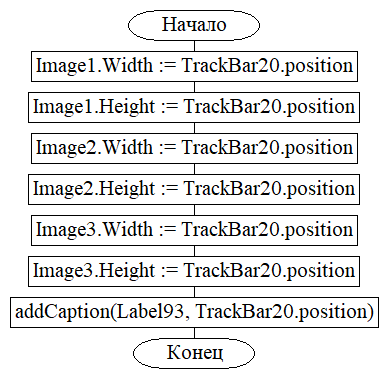
End;

Для всех фигур можно изменить:

* Максимальный шаг вращения, который можно задать ползунком
* Максимальный период вращения, который можно задать ползунком
* Ограничение по длине стороны фигуры/отрезка
* Минимальное значение всех ползунков
* Размер полотна всех фигур

Пояснение: ограничение по длине задается как число A в формуле W-A, где W – размер полотна





*Procedure* TForm1.TrackBar16Change(Sender: TObject);

Begin

maxStep := TrackBar16.position;

TrackBar10.max := maxStep;

TrackBar11.max := maxStep;

TrackBar12.max := maxStep;

addCaptionReal(Label85, TrackBar16.position/1000);

End;

*Procedure* TForm1.TrackBar17Change(Sender: TObject);

Begin

maxTimer := TrackBar17.position;

TrackBar13.max := maxTimer;

TrackBar14.max := maxTimer;

TrackBar15.max := maxTimer;

addCaption(Label87, TrackBar17.position);

End;

*Procedure* TForm1.TrackBar18Change(Sender: TObject);

Begin

limLength := TrackBar18.position;

TrackBar7.max := round((Image1.Height+Image1.Width)/2-limLength);

TrackBar8.max := round((Image2.Height+Image2.Width)/2-limLength);

TrackBar9.max := round((Image3.Height+Image3.Width)/2-limLength);

addCaption(Label89, TrackBar18.position);

End;

*Procedure* TForm1.TrackBar19Change(Sender: TObject);

Begin

minBar := TrackBar19.position;

TrackBar1.min := minBar;

TrackBar2.min := minBar;

TrackBar3.min := minBar;

TrackBar4.min := minBar;

TrackBar5.min := minBar;

TrackBar6.min := minBar;

TrackBar7.min := minBar;

TrackBar8.min := minBar;

TrackBar9.min := minBar;

TrackBar10.min := minBar;

TrackBar11.min := minBar;

TrackBar12.min := minBar;

TrackBar13.min := minBar+1;

TrackBar14.min := minBar+1;

TrackBar15.min := minBar+1;

addCaption(Label91, TrackBar19.position);

End;

*Procedure* TForm1.TrackBar20Change(Sender: TObject);

Begin

Image1.Width := TrackBar20.position;

Image1.Height := TrackBar20.position;

Image2.Width := TrackBar20.position;

Image2.Height := TrackBar20.position;

Image3.Width := TrackBar20.position;

Image3.Height := TrackBar20.position;

addCaption(Label93, TrackBar20.position);

End;

При входе в программу необходимо установить минимумы и максимумы ползунков, а также их начальное положение.

*Procedure* TForm1.FormCreate(Sender: TObject);

Begin

maxStep := 200;

maxTimer := 100;

limLength := 100;

minBar := 0;

CheckBox4.Checked := true;

seg.init(Image1);

reg.init(Image2);

oct.init(Image3);

TrackBar16.min := 100;

TrackBar17.min := 100;

TrackBar18.min := 0;

TrackBar19.min := -100;

TrackBar20.min := 100;

TrackBar16.max := 500;

TrackBar17.max := 500;

TrackBar18.max := 250;

TrackBar19.max := 100;

TrackBar20.max := 700;

TrackBar16.position := 200;

TrackBar17.position := 100;

TrackBar18.position := 100;

TrackBar19.position := 0;

TrackBar20.position := 700;

TrackBar1.min := minBar;

TrackBar2.min := minBar;

TrackBar3.min := minBar;

TrackBar4.min := minBar;

TrackBar5.min := minBar;

TrackBar6.min := minBar;

TrackBar7.min := minBar;

TrackBar8.min := minBar;

TrackBar9.min := minBar;

TrackBar10.min := minBar;

TrackBar11.min := minBar;

TrackBar12.min := minBar;

TrackBar13.min := minBar+1;

TrackBar14.min := minBar+1;

TrackBar15.min := minBar+1;

TrackBar1.max := Image1.Width;

TrackBar2.max := Image2.Width;

TrackBar3.max := Image3.Width;

TrackBar4.max := Image1.Height;

TrackBar5.max := Image2.Height;

TrackBar6.max := Image3.Height;

TrackBar7.max := round((Image1.Height+Image1.Width)/2-limLength);

TrackBar8.max := round((Image2.Height+Image2.Width)/2-limLength);

TrackBar9.max := round((Image3.Height+Image3.Width)/2-limLength);

TrackBar10.max := maxStep;

TrackBar11.max := maxStep;

TrackBar12.max := maxStep;

TrackBar13.max := maxTimer;

TrackBar14.max := maxTimer;

TrackBar15.max := maxTimer;

TrackBar1.position := round(TrackBar1.max/2);

TrackBar2.position := round(TrackBar2.max/2);

TrackBar3.position := round(TrackBar3.max/2);

TrackBar4.position := round(TrackBar4.max/2);

TrackBar5.position := round(TrackBar5.max/2);

TrackBar6.position := round(TrackBar6.max/2);

TrackBar7.position := round(TrackBar7.max/2);

TrackBar8.position := round(TrackBar8.max/2);

TrackBar9.position := round(TrackBar9.max/3);

TrackBar10.position := round(TrackBar10.max/8);

TrackBar11.position := round(TrackBar11.max/8);

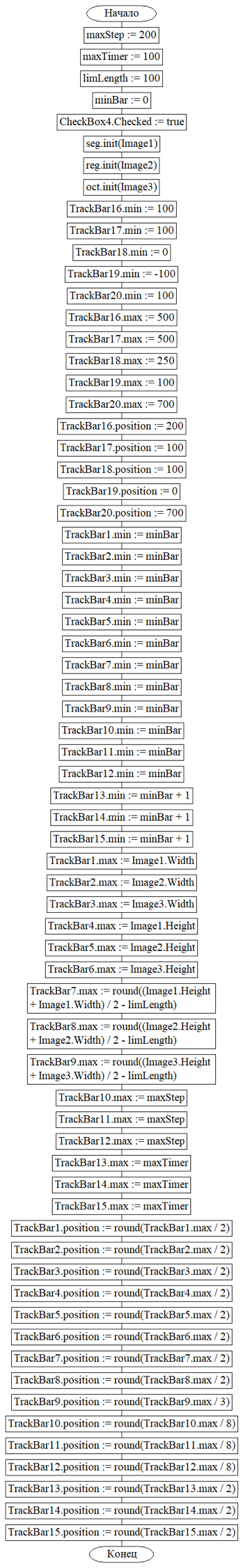
TrackBar12.position := round(TrackBar12.max/8);

TrackBar13.position := round(TrackBar13.max/2);

TrackBar14.position := round(TrackBar14.max/2);

TrackBar15.position := round(TrackBar15.max/2);

End;



Так же я добавил движение фигур в другую сторону. Для этого я создал 3 CheckBox.

*Procedure* TForm1.CheckBox1Change(Sender: TObject);

Var tmp: integer;

Begin

seg.Step := -seg.Step;

tmp := TrackBar10.max;

TrackBar10.max := -TrackBar10.min;

TrackBar10.min := -tmp;

If CheckBox1.Checked Then TrackBar10.position := -round(maxStep/8)

Else TrackBar10.position := round(maxStep/8);

reloadBar(TrackBar10);

End;

*Procedure* TForm1.CheckBox2Change(Sender: TObject);

Var tmp: integer;

Begin

reg.Step := -reg.Step;

tmp := TrackBar11.max;

TrackBar11.max := -TrackBar11.min;

TrackBar11.min := -tmp;

If CheckBox2.Checked Then TrackBar11.position := -round(maxStep/8)

Else TrackBar11.position := round(maxStep/8);

reloadBar(TrackBar11);

End;

*Procedure* TForm1.CheckBox3Change(Sender: TObject);

Var tmp: integer;

Begin

oct.Step := -oct.Step;

tmp := TrackBar12.max;

TrackBar12.max := -TrackBar12.min;

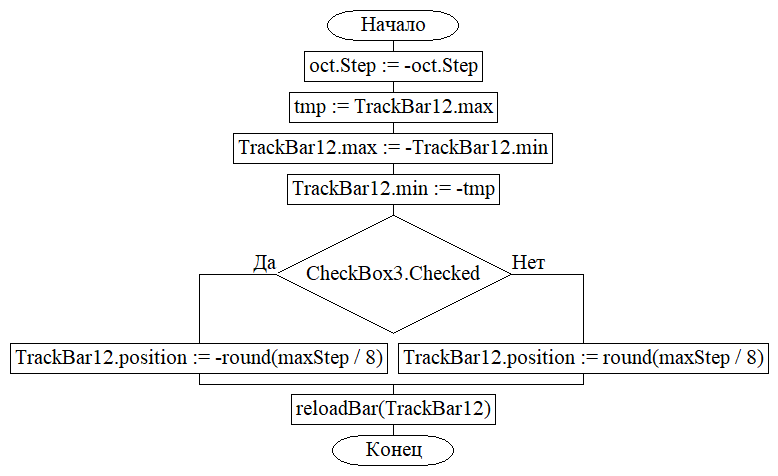
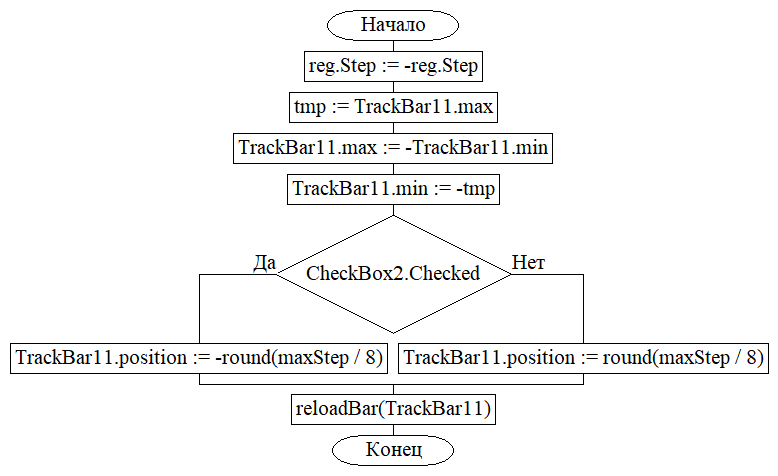
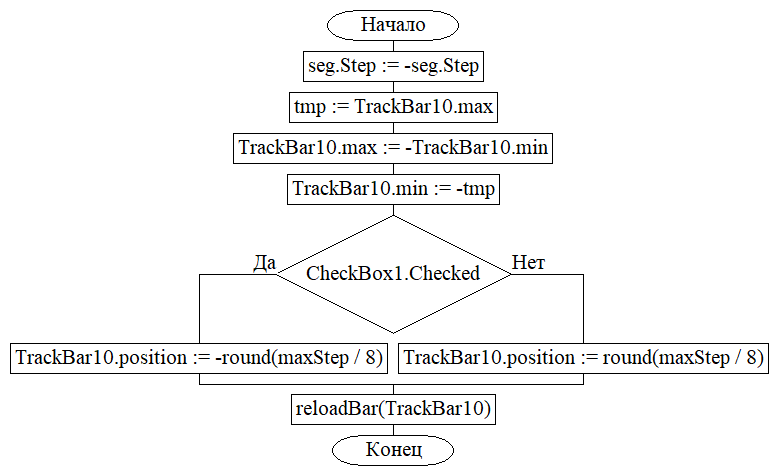
TrackBar12.min := -tmp;

If CheckBox3.Checked Then TrackBar12.position := -round(maxStep/8)

Else TrackBar12.position := round(maxStep/8);

reloadBar(TrackBar12);

End;



Добавил 2 вида цветовых палитр темную и светлую (по умолчанию поставил темную).

*Procedure* TForm1.CheckBox4Change(Sender: TObject);

Begin

If CheckBox4.Checked Then

Begin

Colorbox1.Selected := clWhite;

Colorbox2.Selected := clWhite;

Colorbox3.Selected := clWhite;

Form1.color := $444444;

Form1.font.color := $FFFFFF;

label1.font.color := $FFFFFF;

label4.font.color := $FFFFFF;

label5.font.color := $FFFFFF;

label40.font.color := $FFFFFF;

label41.font.color := $FFFFFF;

label42.font.color := $FFFFFF;

label94.font.color := $FFFFFF;

reloadBar(TrackBar1);

reloadBar(TrackBar2);

reloadBar(TrackBar3);

reloadBar(TrackBar4);

reloadBar(TrackBar5);

reloadBar(TrackBar6);

reloadBar(TrackBar7);

reloadBar(TrackBar8);

reloadBar(TrackBar9);

reloadBar(TrackBar10);

reloadBar(TrackBar11);

reloadBar(TrackBar12);

reloadBar(TrackBar13);

reloadBar(TrackBar14);

reloadBar(TrackBar15);

reloadBar(TrackBar16);

reloadBar(TrackBar17);

reloadBar(TrackBar18);

reloadBar(TrackBar19);

reloadBar(TrackBar20);

ColorBox1.Color := $444444;

ColorBox2.Color := $444444;

ColorBox3.Color := $444444;

End

Else

Begin

Colorbox1.Selected := clBlack;

Colorbox2.Selected := clBlack;

Colorbox3.Selected := clBlack;

Form1.color := clDefault;

Form1.font.color := $000000;

label1.font.color := $000000;

label4.font.color := $000000;

label5.font.color := $000000;

label40.font.color := $000000;

label41.font.color := $000000;

label42.font.color := $000000;

label94.font.color := $000000;

ColorBox1.Color := $FFFFFF;

ColorBox2.Color := $FFFFFF;

ColorBox3.Color := $FFFFFF;

reloadBar(TrackBar1);

reloadBar(TrackBar2);

reloadBar(TrackBar3);

reloadBar(TrackBar4);

reloadBar(TrackBar5);

reloadBar(TrackBar6);

reloadBar(TrackBar7);

reloadBar(TrackBar8);

reloadBar(TrackBar9);

reloadBar(TrackBar10);

reloadBar(TrackBar11);

reloadBar(TrackBar12);

reloadBar(TrackBar13);

reloadBar(TrackBar14);

reloadBar(TrackBar15);

reloadBar(TrackBar16);

reloadBar(TrackBar17);

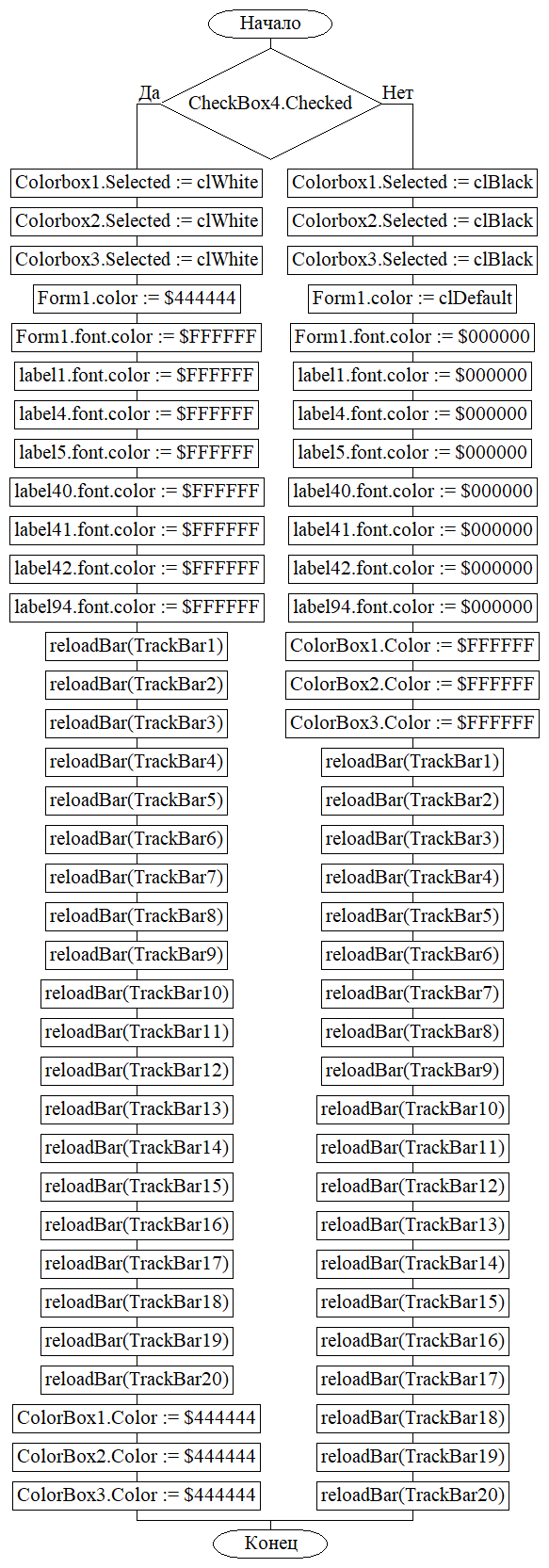
reloadBar(TrackBar18);

reloadBar(TrackBar19);

reloadBar(TrackBar20);

End;

End;



Так же реализовал кнопки сброса и остановки, кнопки выбора цвета

*Procedure* TForm1.Button1Click(Sender: TObject);

Begin

TrackBar10.position := 0;

End;

*Procedure* TForm1.Button2Click(Sender: TObject);

Begin

CheckBox1.Checked := false;

TrackBar1.position := round(TrackBar1.max/2);

TrackBar4.position := round(TrackBar4.max/2);

TrackBar7.position := round(TrackBar7.max/2);

*//TrackBar10.position := round(TrackBar10.max/8);*

TrackBar13.position := round(TrackBar13.max/2);

seg.lastalp := 0;

End;

*Procedure* TForm1.Button3Click(Sender: TObject);

Begin

TrackBar11.position := 0;

End;

*Procedure* TForm1.Button4Click(Sender: TObject);

Begin

CheckBox2.Checked := false;

TrackBar2.position := round(TrackBar1.max/2);

TrackBar5.position := round(TrackBar4.max/2);

TrackBar8.position := round(TrackBar7.max/2);

*//TrackBar11.position := round(TrackBar10.max/8);*

TrackBar14.position := round(TrackBar13.max/2);

reg.lastalp := 0;

End;

*Procedure* TForm1.Button5Click(Sender: TObject);

Begin

TrackBar12.position := 0;

End;

*Procedure* TForm1.Button6Click(Sender: TObject);

Begin

CheckBox3.Checked := false;

TrackBar3.position := round(TrackBar1.max/2);

TrackBar6.position := round(TrackBar4.max/2);

TrackBar9.position := round(TrackBar7.max/2);

*//TrackBar12.position := round(TrackBar10.max/8);*

TrackBar15.position := round(TrackBar13.max/2);

oct.lastalp := 0;

End;

*Procedure* TForm1.Button7Click(Sender: TObject);

Begin

TrackBar16.position := 200;

TrackBar17.position := 100;

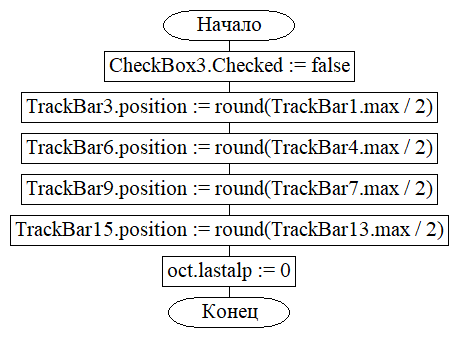
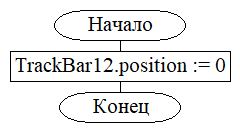
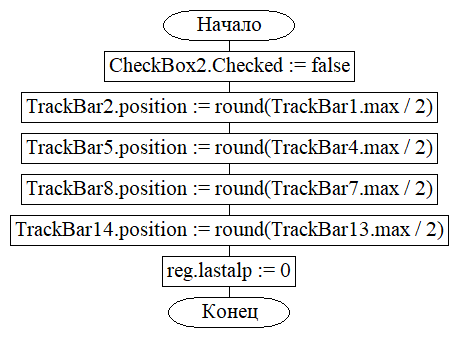
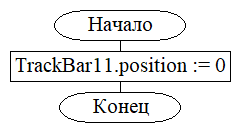
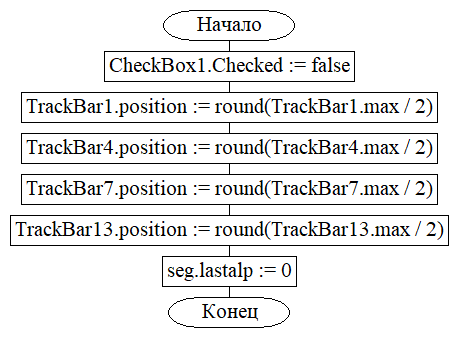
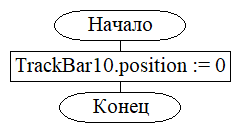
TrackBar18.position := 100;

TrackBar19.position := 0;

TrackBar20.position := 700;

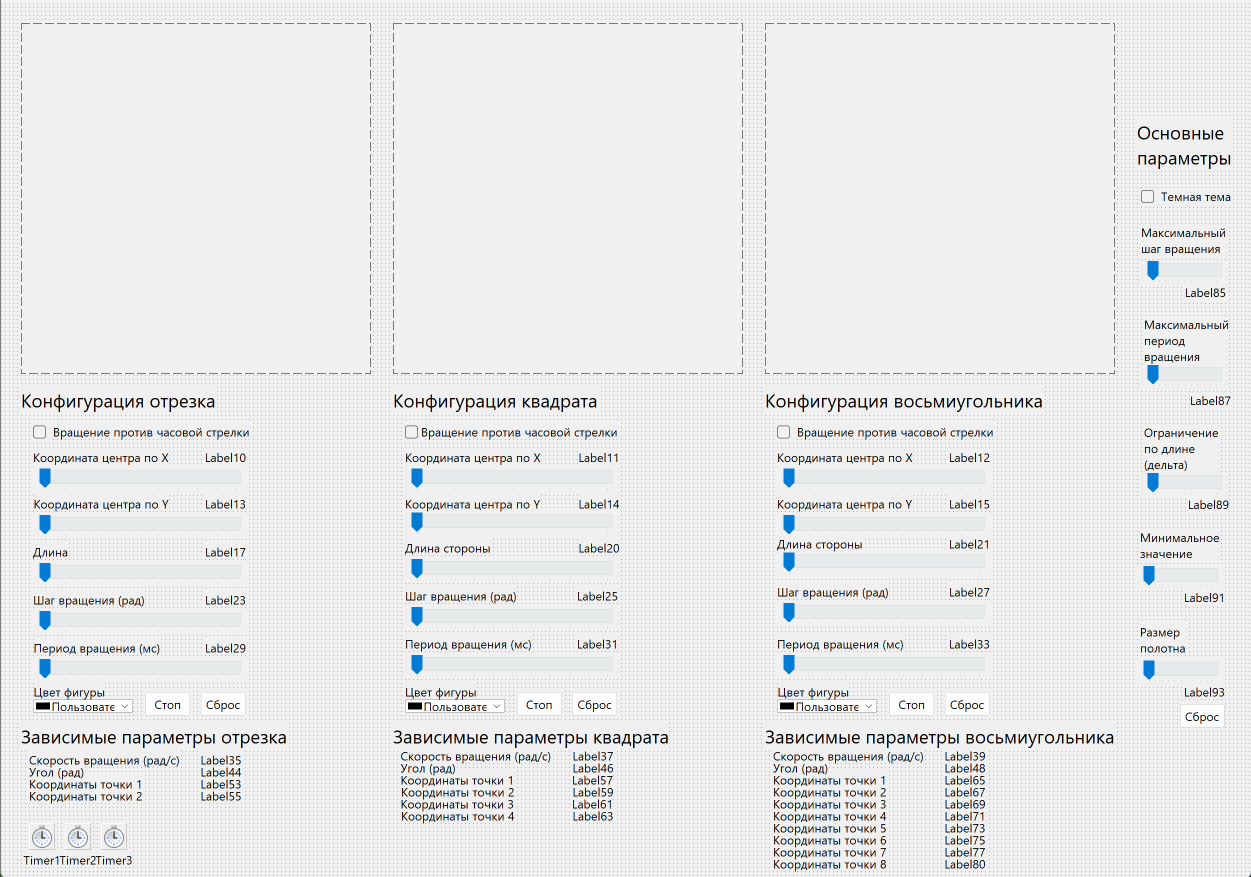
CheckBox4.Checked := true;

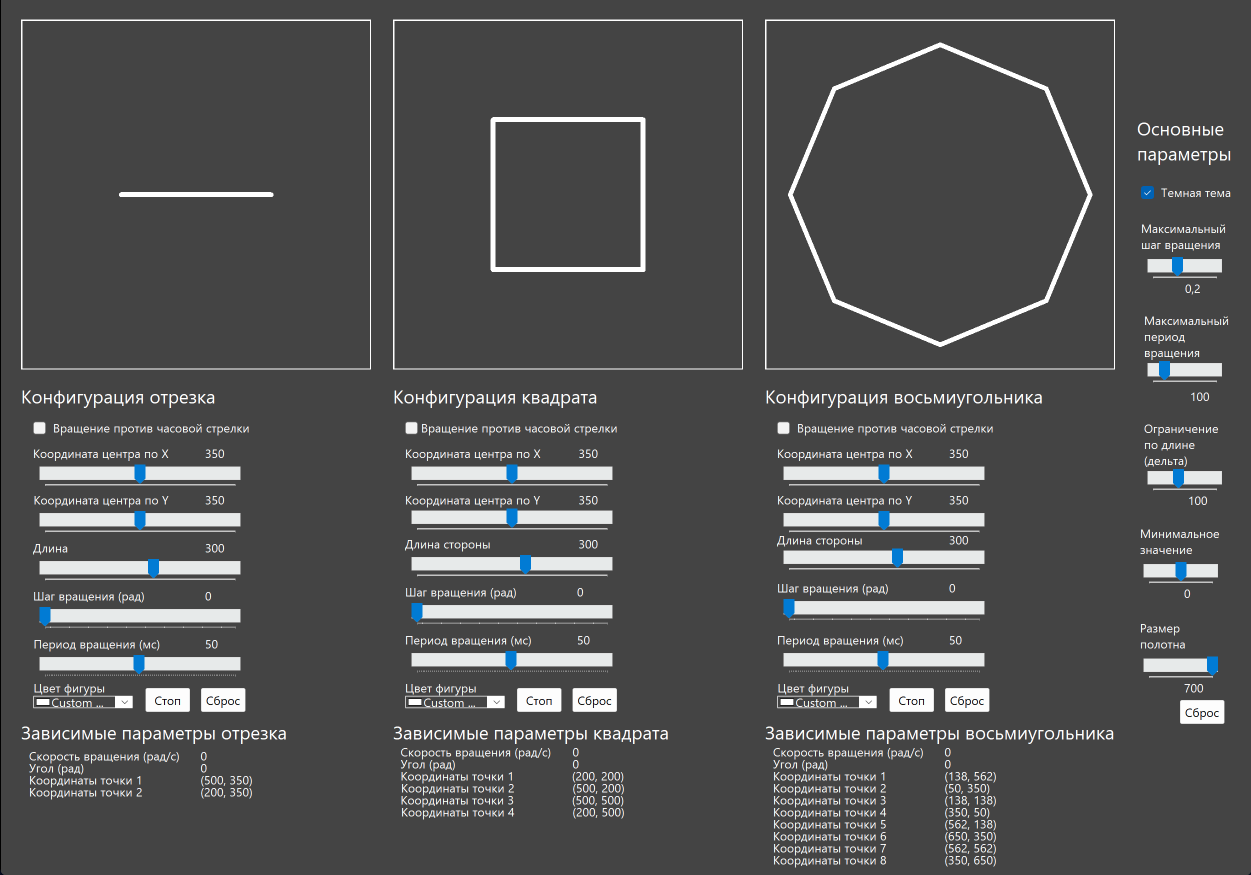
End;

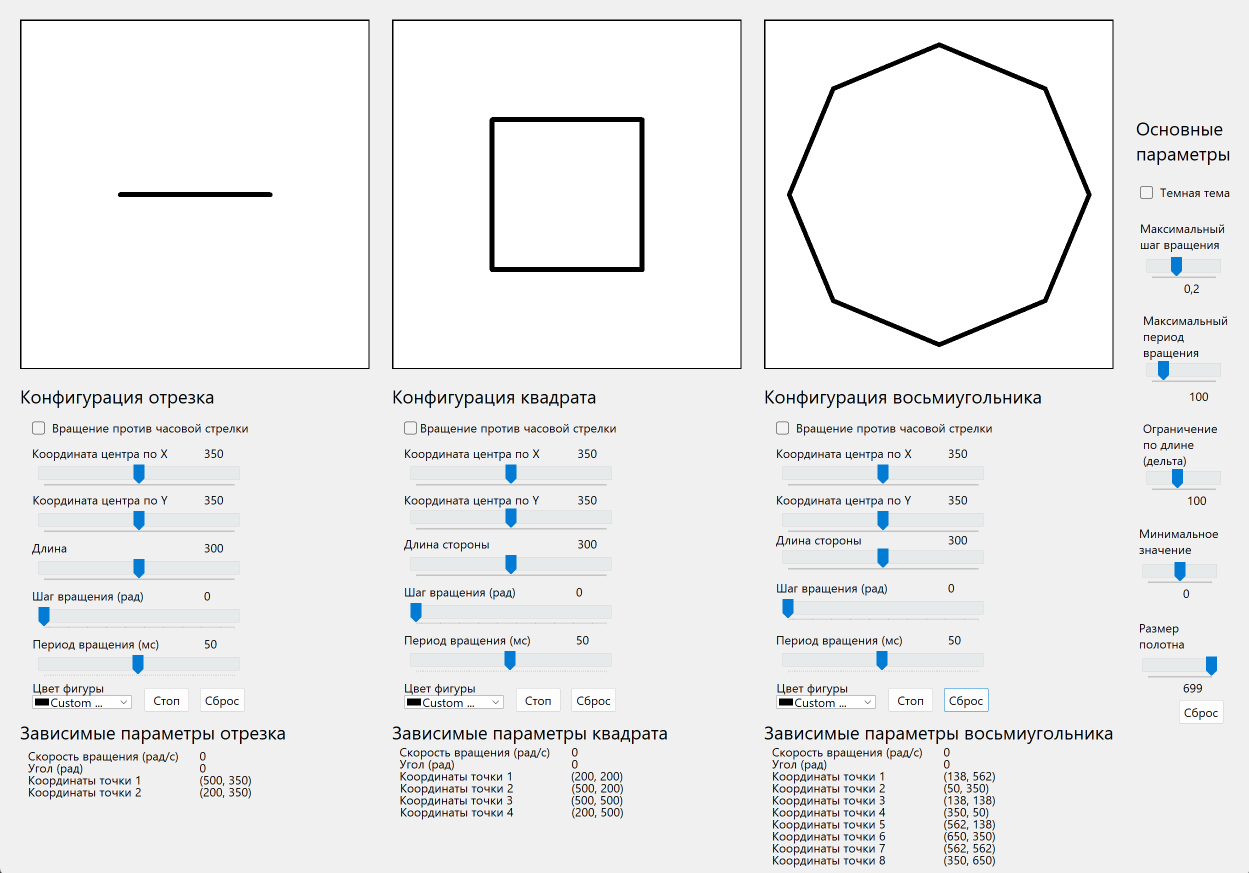


Пояснение: кнопки сброса не влияют на шаг, для того чтобы можно было остановить фигуру с углом в 0 рад.

Ниже представлен шаблон интерфейса, интерфейс (темный и светлый), диаграмма используемых классов Delphi и разработанных классов, граф состояний пользовательского интерфейса и объектную декомпозиция.







Type TForm1 = Class(TForm)

Button1: TButton;

Button2: TButton;

Button3: TButton;

Button4: TButton;

Button5: TButton;

Button6: TButton;

Button7: TButton;

CheckBox1: TCheckBox;

CheckBox2: TCheckBox;

CheckBox3: TCheckBox;

CheckBox4: TCheckBox;

ColorBox1: TColorBox;

ColorBox2: TColorBox;

ColorBox3: TColorBox;

Image1: TImage;

Image2: TImage;

Image3: TImage;

Label1: TLabel;

Label10: TLabel;

Label11: TLabel;

Label12: TLabel;

Label13: TLabel;

Label14: TLabel;

Label15: TLabel;

Label16: TLabel;

Label17: TLabel;

Label18: TLabel;

Label19: TLabel;

Label2: TLabel;

Label20: TLabel;

Label21: TLabel;

Label22: TLabel;

Label23: TLabel;

Label24: TLabel;

Label25: TLabel;

Label26: TLabel;

Label27: TLabel;

Label28: TLabel;

Label29: TLabel;

Label3: TLabel;

Label30: TLabel;

Label31: TLabel;

Label32: TLabel;

Label33: TLabel;

Label34: TLabel;

Label35: TLabel;

Label36: TLabel;

Label37: TLabel;

Label38: TLabel;

Label39: TLabel;

Label4: TLabel;

Label40: TLabel;

Label41: TLabel;

Label42: TLabel;

Label43: TLabel;

Label44: TLabel;

Label45: TLabel;

Label46: TLabel;

Label47: TLabel;

Label48: TLabel;

Label49: TLabel;

Label5: TLabel;

Label50: TLabel;

Label51: TLabel;

Label52: TLabel;

Label53: TLabel;

Label54: TLabel;

Label55: TLabel;

Label56: TLabel;

Label57: TLabel;

Label58: TLabel;

Label59: TLabel;

Label6: TLabel;

Label60: TLabel;

Label61: TLabel;

Label62: TLabel;

Label63: TLabel;

Label64: TLabel;

Label65: TLabel;

Label66: TLabel;

Label67: TLabel;

Label68: TLabel;

Label69: TLabel;

Label7: TLabel;

Label70: TLabel;

Label71: TLabel;

Label72: TLabel;

Label73: TLabel;

Label74: TLabel;

Label75: TLabel;

Label76: TLabel;

Label77: TLabel;

Label78: TLabel;

Label79: TLabel;

Label8: TLabel;

Label80: TLabel;

Label81: TLabel;

Label82: TLabel;

Label83: TLabel;

Label84: TLabel;

Label85: TLabel;

Label86: TLabel;

Label87: TLabel;

Label88: TLabel;

Label89: TLabel;

Label9: TLabel;

Label90: TLabel;

Label91: TLabel;

Label92: TLabel;

Label93: TLabel;

Label94: TLabel;

Timer1: TTimer;

Timer2: TTimer;

Timer3: TTimer;

TrackBar1: TTrackBar;

TrackBar16: TTrackBar;

TrackBar17: TTrackBar;

TrackBar18: TTrackBar;

TrackBar19: TTrackBar;

TrackBar2: TTrackBar;

TrackBar20: TTrackBar;

TrackBar3: TTrackBar;

TrackBar4: TTrackBar;

TrackBar5: TTrackBar;

TrackBar6: TTrackBar;

TrackBar7: TTrackBar;

TrackBar8: TTrackBar;

TrackBar9: TTrackBar;

TrackBar10: TTrackBar;

TrackBar11: TTrackBar;

TrackBar12: TTrackBar;

TrackBar13: TTrackBar;

TrackBar14: TTrackBar;

TrackBar15: TTrackBar;

*Procedure* Button1Click(Sender: TObject);

*Procedure* Button2Click(Sender: TObject);

*Procedure* Button3Click(Sender: TObject);

*Procedure* Button4Click(Sender: TObject);

*Procedure* Button5Click(Sender: TObject);

*Procedure* Button6Click(Sender: TObject);

*Procedure* Button7Click(Sender: TObject);

*Procedure* CheckBox1Change(Sender: TObject);

*Procedure* CheckBox2Change(Sender: TObject);

*Procedure* CheckBox3Change(Sender: TObject);

*Procedure* CheckBox4Change(Sender: TObject);

*Procedure* FormCreate(Sender: TObject);

*Procedure* Timer1Timer(Sender: TObject);

*Procedure* Timer2Timer(Sender: TObject);

*Procedure* Timer3Timer(Sender: TObject);

*Procedure* TrackBar1Change(Sender: TObject);

*Procedure* TrackBar2Change(Sender: TObject);

*Procedure* TrackBar3Change(Sender: TObject);

*Procedure* TrackBar4Change(Sender: TObject);

*Procedure* TrackBar5Change(Sender: TObject);

*Procedure* TrackBar6Change(Sender: TObject);

*Procedure* TrackBar7Change(Sender: TObject);

*Procedure* TrackBar8Change(Sender: TObject);

*Procedure* TrackBar9Change(Sender: TObject);

*Procedure* TrackBar10Change(Sender: TObject);

*Procedure* TrackBar11Change(Sender: TObject);

*Procedure* TrackBar12Change(Sender: TObject);

*Procedure* TrackBar13Change(Sender: TObject);

*Procedure* TrackBar14Change(Sender: TObject);

*Procedure* TrackBar15Change(Sender: TObject);

*Procedure* TrackBar16Change(Sender: TObject);

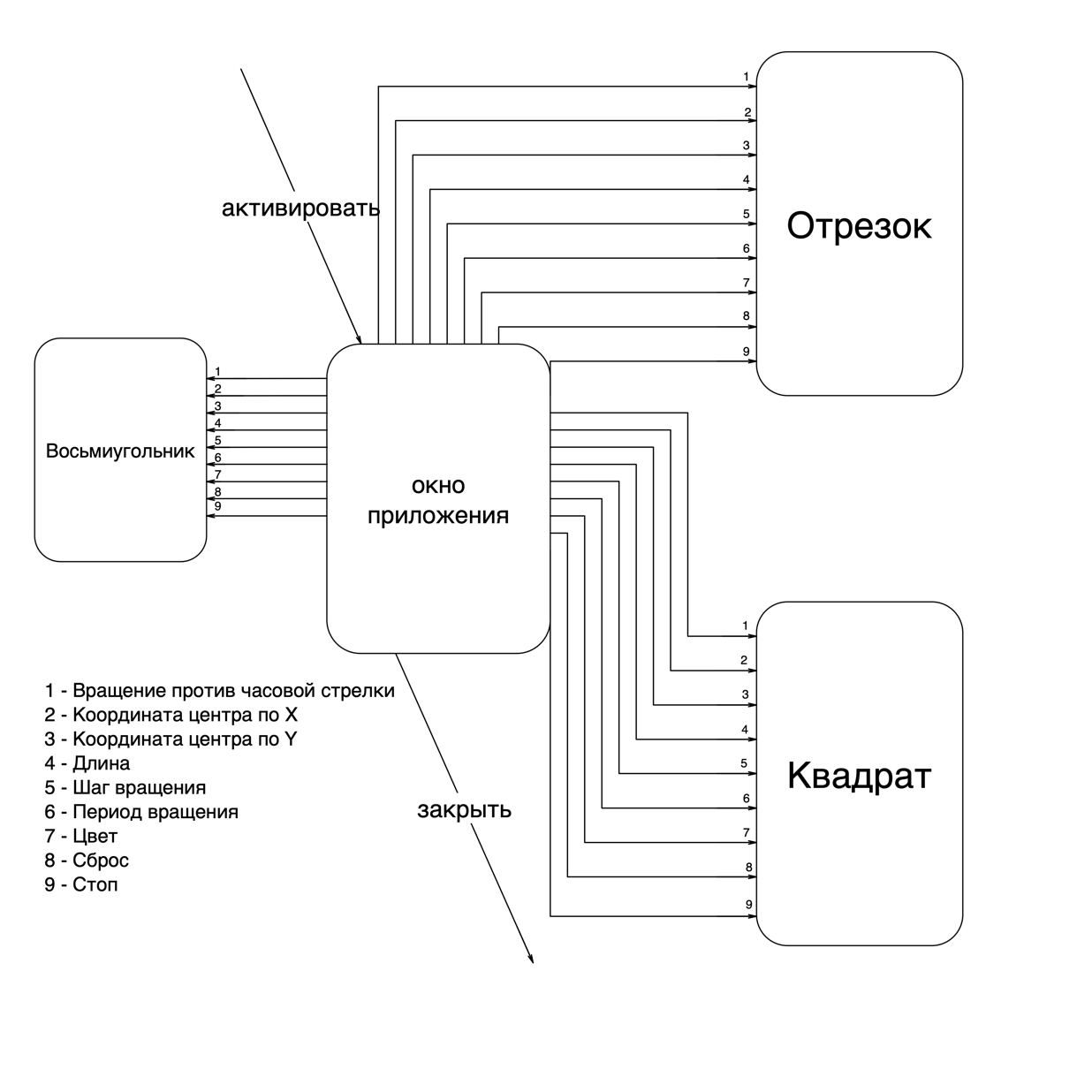
*Procedure* TrackBar17Change(Sender: TObject);

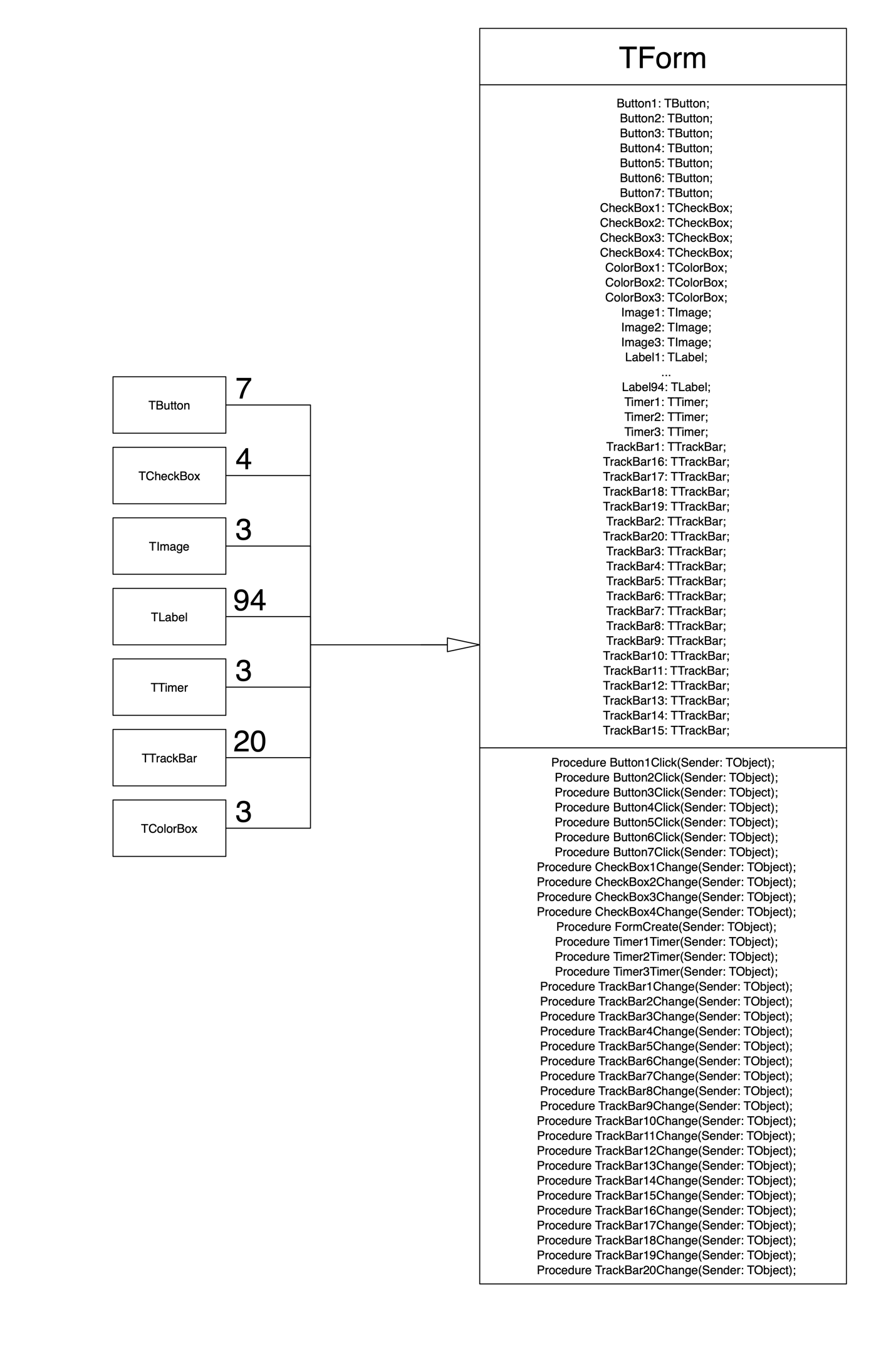
*Procedure* TrackBar18Change(Sender: TObject);

*Procedure* TrackBar19Change(Sender: TObject);

*Procedure* TrackBar20Change(Sender: TObject);

End;



Вывод: Я научился создавать геометрические объекты и управлять ими.

**Задание 2**

Моделировать множество, в качестве элементов которого могут использоваться числа заданного диапазона или символы (стандартный тип "множество" не использовать). Операции: добавление элемента, удаление элемента, печать элементов и проверка вхождения. Создать класс-потомок, который содержит функцию определения мощности множества. Тестировать полученную модель.

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

Решение:

Стоит начать с того, как будут представляться числа: так как в условии сказано, что необходимо реализовать функцию добавления элемента, то при инициализации множества передавать ничего не будем, для хранения элементов будем использовать массив, а чтобы элементы были неповторяющиеся при добавлении элементов будем использовать функцию проверки вхождения, которая тоже необходима по условию. Так как необходимо чтобы множество содержало числа и символы для массива зададим тип Variant, затем для любой работы с элементами массива будем использовать Variant. Вот весь код программы и отладка:

Type TSet = Object

Private

FList: array Of Variant;

Public

*constructor* Create;

*Procedure* Add(AValue: Variant);

*Procedure* Delete(AValue: Variant);

*Function* Contain(AValue: Variant): Boolean;

*Procedure* Print;

End;

*constructor* TSet.Create;

Begin

SetLength(FList, 0);

End;

*Procedure* TSet.Add(AValue: Variant);

Var

I: Integer;

Begin

If Not Contain(AValue) Then

Begin

I := Length(FList);

SetLength(FList, I + 1);

FList[I] := AValue;

End;

End;

*Procedure* TSet.Delete(AValue: Variant);

Var

I, J: Integer;

Begin

I := 0;

*While* I < Length(FList) *Do*

Begin

If FList[I] = AValue Then

Begin

*For* J := I To Length(FList) - 2 *Do*

FList[J] := FList[J + 1];

SetLength(FList, Length(FList) - 1);

End

Else

Inc(I);

End;

End;

*Function* TSet.Contain(AValue: Variant): Boolean;

Var

I: Integer;

Result: Boolean;

Begin

Result := False;

*For* I := 0 To Length(FList) - 1 *Do*

Begin

If FList[I] = AValue Then

Begin

Result := True;

*Break*;

End;

End;

contain := result;

End;

*Procedure* TSet.Print;

Var

I: Integer;

Begin

Write('Elements of the set: ');

*For* I := 0 To Length(FList) - 1 *Do*

Write(FList[I], ' ');

writeln;

End;

Type

TCardinalitySet = Object(TSet)

Public

*Function* Cardinality: Integer;

End;

*Function* TCardinalitySet.Cardinality: Integer;

Begin

Cardinality := Length(FList);

End;

Var PSetChar, PSetInt: TSet;

PCardinalitySet: TCardinalitySet;

Begin

writeln;

PSetChar.create;

PSetChar.add('b');

PSetChar.add('a');

PSetChar.print;

PSetChar.delete('b');

PSetChar.print;

writeln('is b in set: ', PSetChar.contain('b'));

writeln('is a in set: ', PSetChar.contain('a'));

writeln;

PSetInt.create;

PSetInt.add(1);

PSetInt.add(2);

PSetInt.print;

PSetInt.delete(1);

PSetInt.print;

writeln('is 1 in set: ', PSetInt.contain(1));

writeln('is 2 in set: ', PSetInt.contain(2));

writeln;

PCardinalitySet.create;

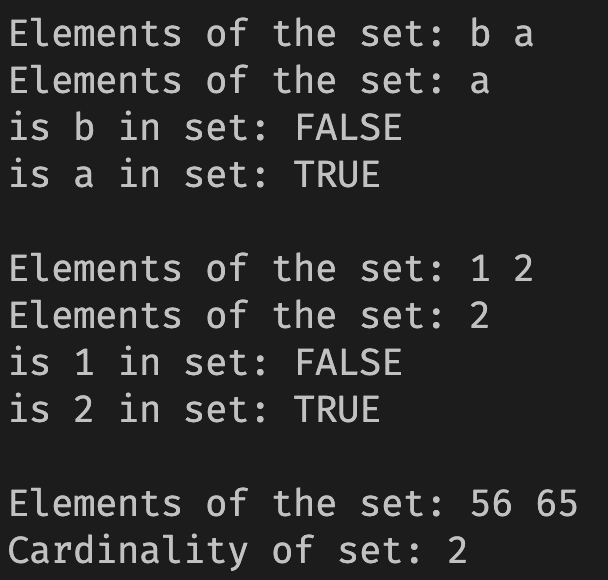
PCardinalitySet.add(56);

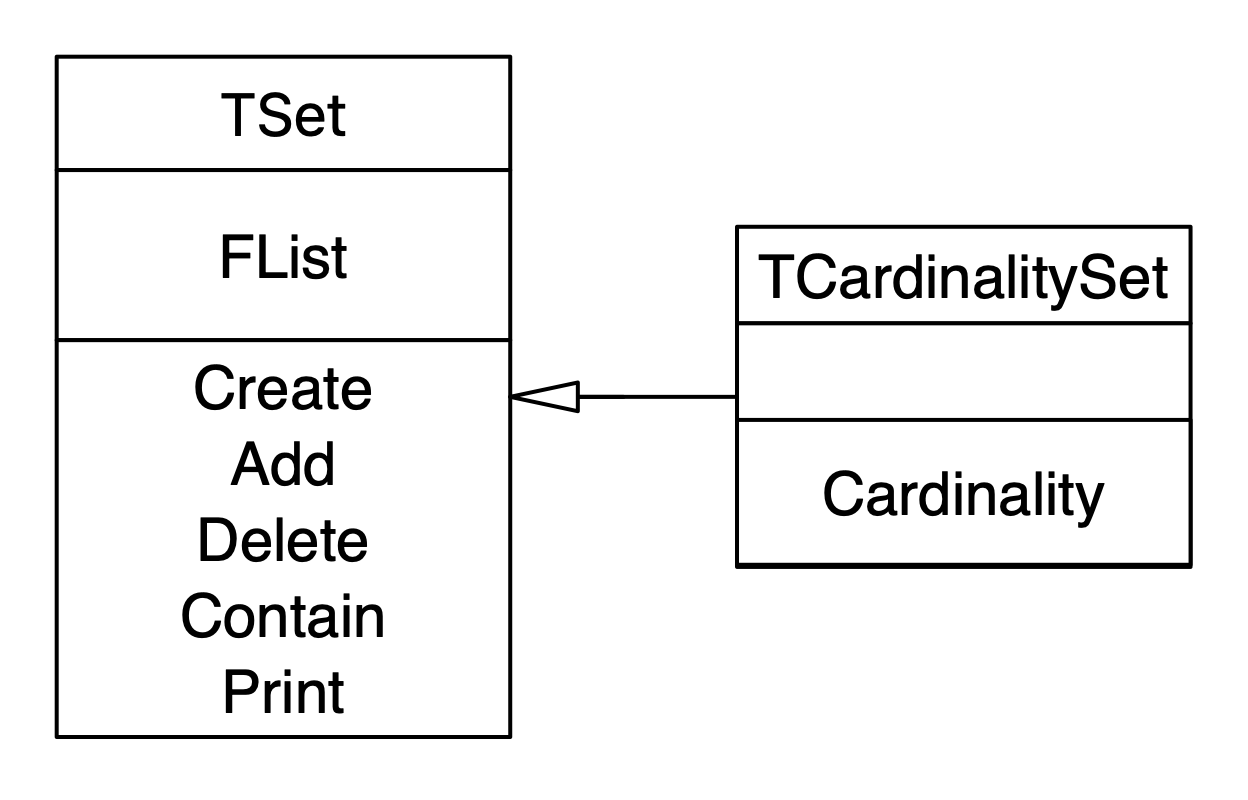
PCardinalitySet.add(65);

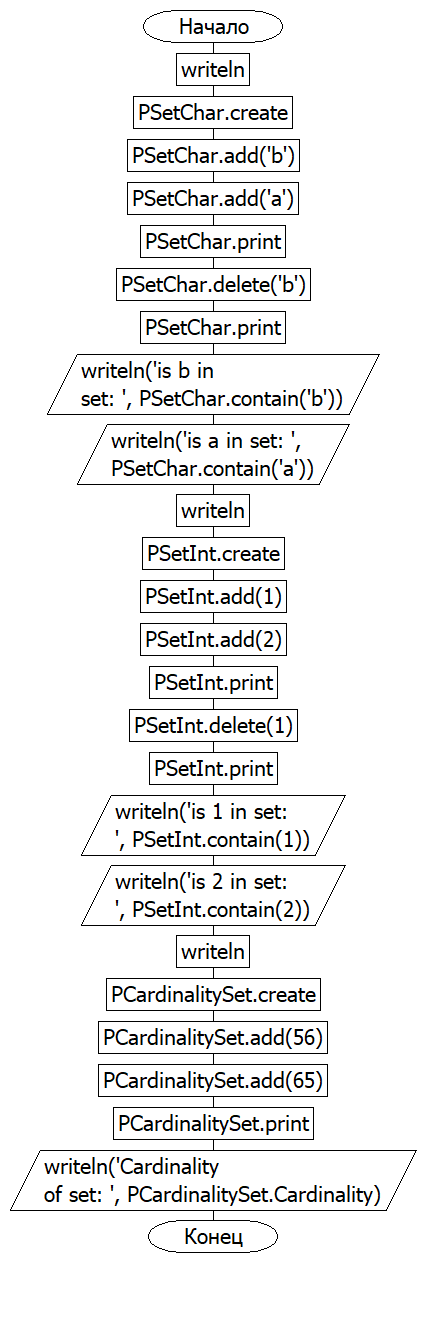
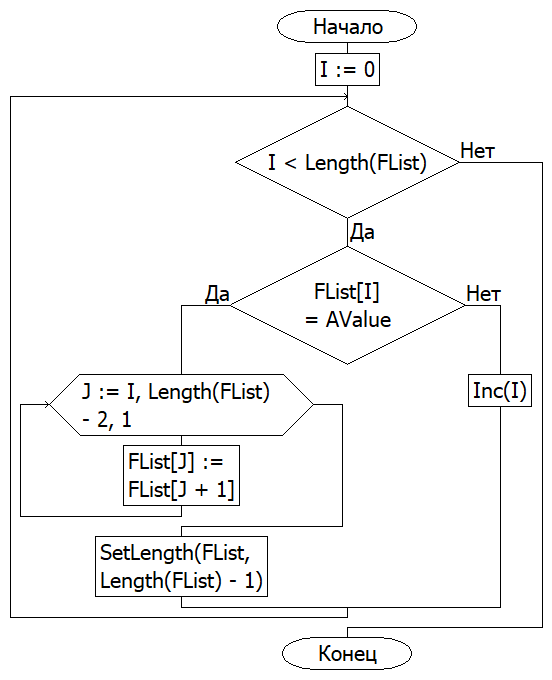
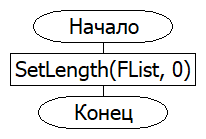
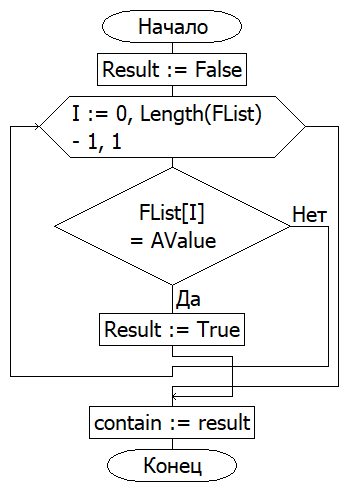
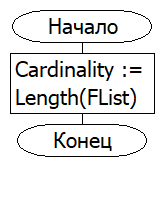
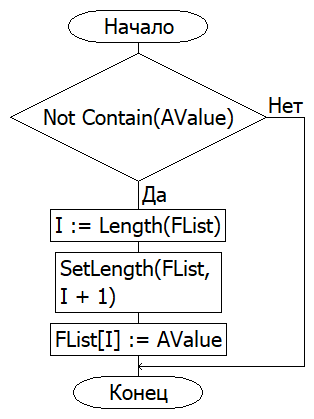
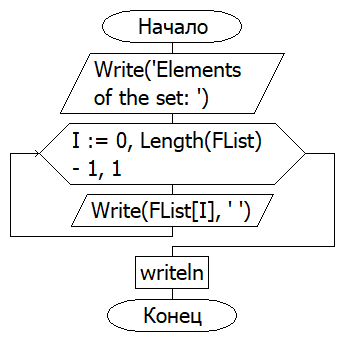
PCardinalitySet.print;

writeln('Cardinality of set: ', PCardinalitySet.Cardinality);

End.







Вывод: я научился создавать класс, который имитирует базовый тип множество.