Министерство образования и науки Российской Федерации Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №10 по курсу «Программирование»

**Разработка графического приложения**

Выполнил студент группы ИВТ-12 /Крючков И. С/ Проверил /Чистяков Г. А./

Киров 2021

**Цель работы:**

# закрепить навыки, полученные в ходе изучения курса.

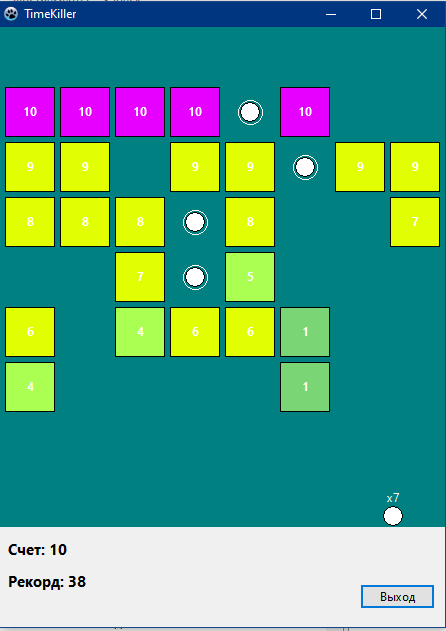
# Задание:

Разработать игру, используя графические возможности среды разработки;

**Схема алгоритма:**

****

**Экранная форма:**



**Листинг кода:**

**Основная программа**

unit main;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls,

Buttons, ComCtrls, Arrow, game, math, Types;

type

{ TForm1 }

TForm1 = class(TForm)

Arrow1: TArrow;

BitBtn1: TBitBtn;

Button1: TButton;

Image1: TImage;

StaticText1: TStaticText;

StaticText2: TStaticText;

StaticText3: TStaticText;

Timer1: TTimer;

Timer2: TTimer;

Timer3: TTimer;

procedure Arrow1Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure FormClose(Sender: TObject; var CloseAction: TCloseAction);

procedure FormCreate(Sender: TObject);

procedure GenLine;

procedure Draw;

procedure Image1MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure Timer1Timer(Sender: TObject);

procedure initBolls;

//procedure drawBlocks;

procedure movingBlocks;

procedure deleteBlock(j: integer);

procedure Timer2Timer(Sender: TObject);

procedure Circle(x\_,y\_,r\_:integer;colour:TColor);

procedure Timer3Timer(Sender: TObject);

//function IsPIn\_Vector(aAx, aAy, aBx, aBy, aCx, aCy, aPx, aPy: single): boolean;

private

public

end;

var

Form1: TForm1;

// Цвета блока в зависимотсти от его массы: 1, 2-5, 6-9, 10-15, 16-25, 25+

Clrs: array of Tcolor = ($74d67a, $52ffab, $00ffe1, $ff00e6, $ffe100, $ff0000);

Blocks: array[0..63] of TBlock;

Bools: array of TBoll;

dx: integer = 5; // Расстояние между блоками по OX

dy: integer = 5; // Расстояние между блоками по OY

bdy: integer = 1; // Отступ шарика от нижней границы

sizeBlock: integer = 50; // Ширина и высота блока

level: integer = 1; //Уровень

newBlocks: boolean = False;

moveBlocks: boolean = False;

offset: integer;

blockSpeed: integer = 5; // Скорость блока

bollSpeed: integer = 8; // Скорость шарика

ziseBoll:integer = 20; // Размер шарика

nb:integer = 1; // Количество шариков

fnb: integer = 0; // Индикаторное колво шариков

sx: integer; // Начальная координата шариков

csx: boolean; // Индикатор значения sx

inb: integer = 1; // Оставшееся кол-во шариков

idb: integer; // Вспомогательная переменная для поочередного вылета шариков

fnd: integer; // Кол-во шариков завершивших движение

gameStarted: boolean;

limzonedy:integer = 30; // Высота нижней границы в которой нельзя запустить шарики

dopBolls: array[0..7] of TDopBoll;

cirRad: integer; // Радиус обводки доп. шарика

dCR: integer = 1; // Изменение радиуса обводки доп. шарика

pnb: integer;

finGame: boolean;

invis:boolean;

recordFile: File of integer;

rec:integer;

implementation

{$R \*.lfm}

{ TForm1 }

procedure TForm1.FormCreate(Sender: TObject);

begin

AssignFile(recordFile, 'record');

try

reset(recordFile);

except

on eInOutError do

begin

rewrite(recordFile);

rec := 0;

end

else

begin

rewrite(recordFile);

rec := 0;

end;

end;

if filesize(recordFile) = 0 then

rec := 0

else

read(recordFile, rec);

if rec < 0 then

rec := 0;

StaticText3.Caption := 'Рекорд: ' + inttostr(rec);

cirRad := ziseBoll div 2 + 1;

Image1.Canvas.Font.Height := 16;

GenLine;

initBolls;

Timer1.Enabled := True;

Draw;

end;

// Повторить

procedure TForm1.BitBtn1Click(Sender: TObject);

var i: integer;

begin

level := 1;

Form1.StaticText1.Caption := 'Счет: ' + inttostr(level);

nb := 1;

inb := 1;

csx := False;

finGame := False;

for i:= 0 to 63 do

Blocks[i] := nil;

for i:= 0 to 7 do

dopBolls[i] := nil;

Image1.Cursor := crDefault;

BitBtn1.Visible := False;

StaticText2.Visible := False;

GenLine;

initBolls;

Timer1.Enabled := True;

Timer3.Enabled := True;

Draw;

end;

// Выход

procedure TForm1.Button1Click(Sender: TObject);

begin

Self.Close;

end;

// При закрытии

procedure TForm1.FormClose(Sender: TObject; var CloseAction: TCloseAction);

begin

if rec < level then

begin

rewrite(recordFile);

write(recordFile, level);

rec := level;

end;

CloseFile(recordFile);

end;

// сброс шариков

procedure TForm1.Arrow1Click(Sender: TObject);

var i:integer;

begin

for i := 0 to nb - 1 do

begin

if Bools[i] <> nil then

begin

Bools[i].dx := 0;

Bools[i].dy := 20;

Bools[i].rbd := True;

invis := True;

end;

end;

end;

// Генерация новых блоков

procedure TForm1.GenLine;

var

i, j, m, n, x, rl, l, t:integer;

xpos: array[0..7] of integer;

begin

// Заполняем массив координат блоков

m := 7;

for i:= 0 to 7 do

begin

// n \* dx + (n-1) \* WitdhBlock

xpos[i] := (i+1) \* dx + i \* sizeBlock;

end;

randomize;

n := random(6)+1; // Количество блоков от 1 до 6

rl := -1;

l := 0;

t := 0;

// Удаляем рандомные координаты блоков, до правильного количества блоков

for i:= 0 to 7-n do

begin

rl := random(8);

// Добавление доп. шариков

while (l < 8) and (t < 1) do

begin

if dopBolls[l] = nil then

begin

dopBolls[l] := TDopBoll.Create(xpos[rl], dy);

dopBolls[l].speed := blockSpeed;

inc(t);

end;

inc(l);

end;

for j := rl to m-1 do

xpos[j] := xpos[j+1];

dec(m);

end;

x := 0;

i := 0;

while (i <= 63) and (x < n) do

begin

if Blocks[i] = nil then

begin

Blocks[i] := TBlock.Create(xpos[x], dy);

Blocks[i].speed := blockSpeed;

Blocks[i].hp := level;

inc(x);

end;

inc(i);

end;

newBlocks := True;

moveBlocks := True;

offset := 0;

end;

procedure TForm1.Draw;

var

i, j, vdx, vdy, k:integer;

p, p1, p2: TRect;

begin

if not finGame then

begin

// Заливка фона

Image1.Canvas.Brush.Color := $808000;

Image1.Canvas.Fillrect(Image1.Canvas.ClipRect);

// Отрисовка блоков

movingBlocks;

// Подпись кол-во шариков

if inb > 0 then

begin

Image1.Canvas.Font.Color := clWhite;

Image1.Canvas.Font.Style := [];

Image1.Canvas.Brush.Style := bsClear;

Image1.Canvas.TextOut(sx - Image1.Canvas.TextWidth('x' + inttostr(inb)) div 2, Image1.Height - bdy - Image1.Canvas.Font.Height - ziseBoll, 'x' + inttostr(inb));

end;

// Летащие шарики

for i := 0 to nb - 1 do

begin

if Bools[i] <> nil then

begin

if Bools[i].n then

begin

if not Bools[i].started then

begin

dec(inb);

Bools[i].started := True;

end;

Bools[i].move;

if not Bools[i].rbd then

begin

vdx := round(bollSpeed\*cos(180\*(arctan2((Bools[i].yn - Bools[i].getY), (Bools[i].xn - Bools[i].getX))/PI + 90)/180\*pi));

if (abs(Bools[i].xn - Bools[i].getX) > abs(vdx)) and (vdx <> 0) then

begin

Bools[i].rbd := True;

Bools[i].ldx := Bools[i].dx;

Bools[i].dx := vdx;

//Memo1.Lines.Add('Xd' + inttostr(abs(Bools[i].xn - Bools[i].getX)) + ' VDX ' + inttostr(abs(Bools[i].xn - (Bools[i].getX + vdx))) + ' XN ' + inttostr(Bools[i].xn) + ' X ' + inttostr(Bools[i].getX) + ' VDX ' + inttostr(vdx));

end;

vdy := round(bollSpeed\*sin(180\*(arctan2((Bools[i].yn - Bools[i].getY), (Bools[i].xn - Bools[i].getX))/PI + 90)/180\*pi));

if (abs(Bools[i].yn - Bools[i].getY) > abs(vdy)) and (vdy <> 0) then

begin

Bools[i].rbd := True;

Bools[i].ldy := Bools[i].dy;

Bools[i].dy := vdy;

end;

end;

end;

Image1.Canvas.Brush.Color := clWhite;

// Отрисовываем шарик

if Bools[i].vis then

begin

Image1.Canvas.Ellipse(Bools[i].getX - ziseBoll div 2, Bools[i].getY - ziseBoll div 2, Bools[i].getX + ziseBoll div 2, Bools[i].getY + ziseBoll div 2);

end;

// Отрисовка первого шарика в новом положении перед стартом

if csx then

begin

Image1.Canvas.Ellipse(sx - ziseBoll div 2, Image1.Height - ziseBoll - bdy, sx + ziseBoll div 2, Image1.Height - bdy);

end;

// хитбокс шарика

p1.Left := Bools[i].getX - ziseBoll div 2;

p1.Top := Bools[i].getY - ziseBoll div 2;

p1.Right := Bools[i].getX + ziseBoll div 2;

p1.Bottom := Bools[i].getY + ziseBoll div 2;

if not invis then

begin

// Проверка на столкновение с блоками и стенками

for j := 0 to 63 do

begin

if (Blocks[j] <> nil) then

begin

// Хитбокс блока

p2.Left := Blocks[j].getX;

p2.Top := Blocks[j].getY;

p2.Right := Blocks[j].getX + sizeBlock;

p2.Bottom := Blocks[j].getY + sizeBlock;

// При пересечении шарика и блока

if intersectrect(p, p1, p2) then

begin

Bools[i].rbd := true;

// Удар снизу

if (p1.Top <= p2.Bottom) and (p1.Top + abs(Bools[i].ldy) >= p2.Bottom) and (Bools[i].lb <> j) and (Bools[i].dy < 0) then

begin

Bools[i].lb := j; // ид блока

Bools[i].setY(p2.Bottom+1); // Выносим шарик за пределы блока

Bools[i].dy := (-1) \* Bools[i].dy;

dec(Blocks[j].hp);

if Blocks[j].hp < 1 then

deleteBlock(j);

end

// Удар сверху

else if (p1.Bottom >= p2.Top) and (p1.Bottom - abs(Bools[i].ldy) <= p2.Top) and (Bools[i].lb <> j) and (Bools[i].dy > 0) then

begin

Bools[i].lb := j;

Bools[i].setY(p2.Top-1);

Bools[i].dy := (-1) \* Bools[i].dy;

dec(Blocks[j].hp);

if Blocks[j].hp < 1 then

deleteBlock(j);

end

// Удар слева

else if (p1.Right >= p2.Left) and (p1.Right - abs(Bools[i].ldx) <= p2.Left) and (Bools[i].lb <> j) and (Bools[i].dx > 0) then

begin

Bools[i].lb := j;

Bools[i].setX(p2.Left-1);

Bools[i].dx := (-1) \* Bools[i].dx;

dec(Blocks[j].hp);

if Blocks[j].hp < 1 then

deleteBlock(j);

end

// Удар справа

else if (p1.Left <= p2.Right) and (p1.Left + abs(Bools[i].ldx) >= p2.Right) and (Bools[i].lb <> j) and (Bools[i].dx < 0) then

begin

Bools[i].lb := j;

Bools[i].setX(p2.Right+1);

Bools[i].dx := (-1) \* Bools[i].dx;

dec(Blocks[j].hp);

if Blocks[j].hp < 1 then

deleteBlock(j);

end;

end;

end;

end;

// Столкновение с доп. шариком

for j := 0 to 7 do

begin

if (dopBolls[j] <> nil) then

begin

// Хитбокс доп. шарика

p2.Left := dopBolls[j].getX + ((sizeBlock - ziseBoll) div 2);

p2.Top := dopBolls[j].getY + ((sizeBlock - ziseBoll) div 2);

p2.Right := dopBolls[j].getX + ((sizeBlock - ziseBoll) div 2) + ziseBoll;

p2.Bottom := dopBolls[j].getY + ((sizeBlock - ziseBoll) div 2)+ ziseBoll;

if intersectrect(p, p1, p2) then

begin

inc(pnb); // увеличение кол-ва шариков.

// Удаление доп. шарика

for k := j to 6 do

dopBolls[k] := dopBolls[k+1];

end;

end;

end;

end;

// Удар о левую стенку

if (p1.Left <= 0) and (Bools[i].lb <> -2)and (Bools[i].dx < 0) then

begin

Bools[i].lb := -2;

Bools[i].rbd := true;

Bools[i].dx := (-1) \* Bools[i].dx;

end

// Удар о правую стенку

else if (p1.Right >= Image1.Width) and (Bools[i].lb <> -3) and (Bools[i].dx > 0) then

begin

Bools[i].lb := -3;

Bools[i].rbd := true;

Bools[i].dx := (-1) \* Bools[i].dx;

end

// Удар о верхнюю стенку

else if (p1.Top <= 0) and (Bools[i].lb <> -4) and (Bools[i].dy < 0) then

begin

Bools[i].lb := -4;

Bools[i].rbd := true;

Bools[i].dy := (-1) \* Bools[i].dy;

end

// Удар о нижнюю стенку (завершение)

else if(p1.Bottom >= Image1.Height) then

begin

// Если новое значение sx не устанвлено, устанавливаем

if not csx then

begin

sx := Bools[i].getX;

csx := True;

end;

Bools[i].n := False; // Не двигаем блок

Bools[i].vis := False; // Далем блок невидимым

if not Bools[i].finished then

begin

// Увеличиваем количество шариков завершивших движение

inc(fnd);

Bools[i].finished := True;

end;

// Если все шарики завершили движение

if fnd = nb then

begin

Timer2.Enabled := False;

inc(level);

Form1.StaticText1.Caption := 'Счет: ' + inttostr(level);

nb := nb + pnb;

pnb := 0;

inb := nb; // Счетчик шариков = кол-во шариков

initBolls;

GenLine; // Генерируем новые блоки

//newBlocks := True; // Отрисовываем новые блоки

moveBlocks := True; // Смещаем блоки

offset := 0;

fnd := 0;

Arrow1.Visible := False;

end;

end;

end;

end;

// Если проиграли

end

else

begin

Timer1.Enabled := False;

Timer2.Enabled := False;

Image1.Cursor := crNoDrop;

BitBtn1.Visible := True;

StaticText2.Visible := True;

if rec < level then

begin

rewrite(recordFile);

write(recordFile, level);

rec := level;

StaticText3.Caption := 'Рекорд: ' + inttostr(rec);

end;

end;

end;

procedure TForm1.Image1MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var i:integer;

begin

// Если игра еще не начата

if not gameStarted and not finGame then

begin

if Y < Image1.Height - limzonedy then

begin

for i := 0 to nb - 1 do

begin

if Bools[i] <> nil then

begin

Bools[i].setX(sx);

Bools[i].setY(Image1.Height - ziseBoll div 2 - bdy);

fnb := 0;

Bools[i].xn := X;

Bools[i].yn := Y;

Bools[i].dx := round(bollSpeed\*cos(180\*(arctan2((Y - Bools[i].getY), (X - Bools[i].getX))/PI + 90)/180\*pi));

Bools[i].dy := round(bollSpeed\*sin(180\*(arctan2((Y - Bools[i].getY), (X - Bools[i].getX))/PI + 90)/180\*pi));

if i = 0 then

Bools[i].n := True;

Bools[i].vis := True;

Bools[i].started := False;

Bools[i].finished := False;

Bools[i].lb := -1;

Bools[i].ldx := Bools[i].dx;

Bools[i].ldy := Bools[i].dy;

end;

end;

fnd := 0;

csx := False;

inb := nb;

idb := 1;

gameStarted := True;

Timer2.Enabled := True;

Timer1.Enabled := True;

Arrow1.Visible := True;

invis := False;

end;

end;

end;

procedure TForm1.Timer1Timer(Sender: TObject);

begin

Draw;

end;

procedure TForm1.initBolls;

var i: integer;

begin

SetLength(Bools, nb);

for i := 0 to nb - 1 do

begin

if not csx then

sx := Image1.Width div 2;

Bools[i] := TBoll.Create(sx, Image1.Height - ziseBoll div 2 - bdy);

idb := 1;

Bools[i].xn := 0;

Bools[i].yn := 0;

Bools[i].dx := 0;

Bools[i].dy := 0;

Bools[i].vis := True;

Bools[i].started := False;

Bools[i].finished := False;

Bools[i].n := False; // Активность шарика.

Bools[i].rbd := False

end;

end;

procedure TForm1.Timer2Timer(Sender: TObject);

var i:integer;

begin

for i := idb to nb - 1 do

begin

if Bools[i] <> nil then

begin

Bools[i].n := True;

idb := i + 1;

break;

end;

end;

end;

// Move blocks

procedure TForm1.movingBlocks;

var i: integer;

blockColor: Tcolor;

begin

for i:= 0 to 63 do

begin

if (Blocks[i] <> nil) then

begin

if Blocks[i].hp > 0 then

begin

if offset <> sizeBlock + dy then

begin

if moveBlocks then

Blocks[i].move

end;

case Blocks[i].hp of

1: blockColor := Clrs[0];

2..5: blockColor := Clrs[1];

6..9: blockColor := Clrs[2];

10..15: blockColor := Clrs[3];

16..25: blockColor := Clrs[4];

else blockColor := Clrs[5];

end;

Image1.Canvas.Brush.Color := blockColor;

Image1.Canvas.rectangle(Blocks[i].getX, Blocks[i].getY, Blocks[i].getX+sizeBlock, Blocks[i].getY+sizeBlock);

// Вывод hp каждого блока в центре

Image1.Canvas.Font.Color := clWhite;

Image1.Canvas.Font.Style := [fsbold];

Image1.Canvas.Brush.Style := bsClear;

Image1.Canvas.TextOut(Blocks[i].getX + sizeBlock div 2 - Image1.Canvas.TextWidth(inttostr(Blocks[i].hp)) div 2, Blocks[i].getY + sizeBlock div 2 - Image1.Canvas.Font.Height div 2, inttostr(Blocks[i].hp));

if Blocks[i].getY >= Image1.Height - (sizeBlock + dy) then

finGame := True;

end;

end;

end;

for i := 0 to 7 do

begin

if dopBolls[i] <> nil then

begin

if offset <> sizeBlock + dy then

begin

if moveBlocks then

dopBolls[i].move

end

else

begin

Timer1.Enabled := False;

moveBlocks := False;

gameStarted := False;

end;

Image1.Canvas.Brush.Color := clWhite;

Image1.Canvas.Ellipse(dopBolls[i].getX + ((sizeBlock - ziseBoll) div 2), dopBolls[i].getY + ((sizeBlock - ziseBoll) div 2), dopBolls[i].getX + ((sizeBlock - ziseBoll) div 2) + ziseBoll, dopBolls[i].getY + ((sizeBlock - ziseBoll) div 2) + ziseBoll);

Circle(dopBolls[i].getX + sizeBlock div 2, dopBolls[i].getY + sizeBlock div 2, cirRad, clWhite);

end;

end;

offset := offset + blockSpeed;

end;

procedure TForm1.deleteBlock(j: integer);

var i:integer;

begin

for i := j to 62 do

Blocks[i] := Blocks[i+1];

end;

// Рисование круга

procedure TForm1.Circle(x\_,y\_,r\_:integer;colour:TColor);

var x,y,d1,d2:integer;

begin

x:=0;

y:=r\_;

while ( x<=round(r\_/sqrt(2)) )

do begin

Form1.Image1.Canvas.Pixels[x\_+x,y\_+y]:=colour;

Form1.Image1.Canvas.Pixels[x\_+x,y\_-y]:=colour;

Form1.Image1.Canvas.Pixels[x\_-x,y\_+y]:=colour;

Form1.Image1.Canvas.Pixels[x\_-x,y\_-y]:=colour;

Form1.Image1.Canvas.Pixels[x\_+y,y\_+x]:=colour;

Form1.Image1.Canvas.Pixels[x\_+y,y\_-x]:=colour;

Form1.Image1.Canvas.Pixels[x\_-y,y\_+x]:=colour;

Form1.Image1.Canvas.Pixels[x\_-y,y\_-x]:=colour;

x:=x+1;

d1:=ABS(r\_\*r\_-x\*x-y\*y);

d2:=ABS(r\_\*r\_-x\*x-(y-1)\*(y-1));

if(d1>d2) then y:=y-1;

end;

end;

procedure TForm1.Timer3Timer(Sender: TObject);

var i:integer;

begin

if cirRad = ziseBoll div 2 + 5 then

dCR := -1

else if cirRad = ziseBoll div 2 + 1 then

dCR := 1;

if Timer1.Enabled then

cirRad := cirRad + dCR

else

begin

for i := 0 to 7 do

begin

if dopBolls[i] <> nil then

begin

//Зарисоваваем уже нарисованное цветом фона

Image1.Canvas.Brush.Color := $808000;

Image1.Canvas.Ellipse(dopBolls[i].getX + ((sizeBlock - ziseBoll) div 2), dopBolls[i].getY + ((sizeBlock - ziseBoll) div 2), dopBolls[i].getX + ((sizeBlock - ziseBoll) div 2) + ziseBoll, dopBolls[i].getY + ((sizeBlock - ziseBoll) div 2) + ziseBoll);

Circle(dopBolls[i].getX + sizeBlock div 2, dopBolls[i].getY + sizeBlock div 2, cirRad, $808000);

end;

end;

cirRad := cirRad + dCR;

for i := 0 to 7 do

begin

if dopBolls[i] <> nil then

begin

Image1.Canvas.Brush.Color := clWhite;

Image1.Canvas.Ellipse(dopBolls[i].getX + ((sizeBlock - ziseBoll) div 2), dopBolls[i].getY + ((sizeBlock - ziseBoll) div 2), dopBolls[i].getX + ((sizeBlock - ziseBoll) div 2) + ziseBoll, dopBolls[i].getY + ((sizeBlock - ziseBoll) div 2) + ziseBoll);

Circle(dopBolls[i].getX + sizeBlock div 2, dopBolls[i].getY + sizeBlock div 2, cirRad, clWhite);

end;

end;

end;

end;

end.

**Модуль game**

unit game;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Graphics;

type

TBlock = class (TObject)

private

x, y: integer;

//speed\_H: Integer;

public

speed: integer;

color: Tcolor;

hp: integer;

Constructor Create(x0, y0: Integer);

procedure move;

function getX: Integer;

function getY: Integer;

end;

TBoll = class (TObject)

private

x, y: integer;

public

xn, yn, dx, dy, lb, ldx, ldy: integer;

n, rbd: boolean;

vis, started, finished: boolean;

Constructor Create(x0, y0: Integer);

procedure move;

function getX: Integer;

function getY: Integer;

procedure setX(xsn:integer);

procedure setY(ysn:integer);

end;

TDopBoll = class (TObject)

private

x, y: integer;

public

speed: integer;

Constructor Create(x0, y0: Integer);

procedure move;

function getX: Integer;

function getY: Integer;

end;

implementation

Constructor TBlock.Create(x0, y0: Integer);

begin

x := x0;

y := y0;

end;

procedure TBlock.move;

begin

y := y + speed;

end;

function TBlock.getX: integer;

begin

Result := x;

end;

function TBlock.getY: integer;

begin

Result := y;

end;

Constructor TBoll.Create(x0, y0: Integer);

begin

x := x0;

y := y0;

end;

procedure TBoll.move;

begin

x := x + dx;

y := y + dy;

end;

function TBoll.getX: integer;

begin

Result := x;

end;

function TBoll.getY: integer;

begin

Result := y;

end;

procedure TBoll.setX(xsn:integer);

begin

x := xsn;

end;

procedure TBoll.setY(ysn:integer);

begin

y := ysn;

end;

Constructor TDopBoll.Create(x0, y0: Integer);

begin

x := x0;

y := y0;

end;

procedure TDopBoll.move;

begin

y := y + speed;

end;

function TDopBoll.getX: integer;

begin

Result := x;

end;

function TDopBoll.getY: integer;

begin

Result := y;

end;

end.

**Вывод:** в ходе выполнения лабораторной работы были получены навыки работы с графиков в среде разработки Lazarus, закреплены навыки, полученные в ходе изучения курса.