Тема: Статические методы

Задание:

 В модуль, созданный в лабораторной работе №3, добавить методы движения фигур (в каждый класс). Использовать статические методы. Написать программу, позволяющую выбирать фигуру для движения и вид движения: случайный или с помощью стрелок. Подключить к этой программе созданный модуль с описанием графических классов.

в основной программе попробуйте вместо операторов if..then..else использовать операторы Case.

Листинг файла LABA 4.pas

*uses crt, graph, figura;*

*procedure coords(a:char; var NewX, NewY:integer; h:integer);*

*begin*

*case a of*

*#80: begin inc(NewY,h); if NewY>GetMaxY-20 then NewY:=20;*

 *end;*

*#75: begin dec(NewX,h); if NewX<10 then NewX:=GetMaxX-10;*

 *end;*

*#77: begin inc(NewX,h); if NewX>GetMaxX-10 then NewX:=10;*

 *end;*

*#72: begin dec(NewY,h); if NewY<20 then NewY:=GetMaxY-20;*

 *end;*

*end;*

*end;*

*var p:tPoint;*

 *l: tLine;*

 *t:tTriang;*

 *r:tRectang;*

 *c:tCircle;*

 *e:tEllipse;*

 *i:integer;*

 *grDriver, grMode, ErrCode :integer;*

 *grPath:string;*

 *xv,yv,xvk,yvk,xvk2,yvk2 : integer;*

 *colg : word;*

 *a:char;*

 *NewX,NewY,NewX2,NewY2,NewX3,NewY3,h:integer;*

 *rej,Q,fig:string;*

*begin*

*grDriver:=VGA;*

*grMode:=VGAHi;*

*grPath:='d:\tp\bgi';*

*InitGraph(grDriver, grMode, grPath);*

*ErrCode:=GraphResult;*

*if ErrCode<> grOk then*

*begin*

*writeln('Oshibka init graf rejima');*

*readln;*

*Halt(1);*

*end;*

*RestoreCrtMode;*

*writeln('Graph rejim zagruj yspeshno');*

*Q:='n';*

*xv:=100; yv:=100;*

*xvk:=200; yvk:=300;*

*xvk2:=50; yvk2:=300;*

*h:=10;*

*while Q<>'y' do*

*begin*

*writeln('Viberite rejim dvij tochki:');*

*writeln('s-sluchain dvij');*

*writeln('pr - pryam. dvij');*

*readln(rej);*

*writeln('Viberite figury:');*

*writeln('po-tochka');*

*writeln('o-okrujn');*

*writeln('e-ellipse');*

*writeln('l-liniya');*

*writeln('t-treugolnik');*

*writeln('p-prymougolnik');*

*readln(fig);*

*SetGraphMode(grMode);*

*if fig='po' then*

*begin*

*p.Init(xv,yv,10,9);*

*p.Draw;*

*end else*

*if fig='o' then*

*begin*

*c.Init(xv,yv+150,10,10,50);*

*c.Draw;*

*end else*

*if fig='e' then*

*begin*

*e.Init(100,100,10,9,20,50,0,360);*

*e.Draw;*

*end else*

*if fig='l' then*

*begin*

*l.Init(xv,yv,10,13,xvk,yvk);*

*l.Draw;*

*end else*

*if fig='t' then*

*begin*

*t.Init(xv,yv,10,12,xv+50,yv+50,xv-50,yv+70);*

*t.Draw;*

*end else*

*if fig='p' then*

*begin*

*r.Init(xv,yv,10,11,xv+100,yv+50);*

*r.Draw;*

*end else*

*begin*

*writeln('Error!');*

*end;*

*repeat*

*a:=readkey;*

*if rej='pr'then begin*

*if fig='po' then*

*begin*

*NewX:=p.GetX;*

*NewY:=p.GetY;*

*Coords(a,NewX,NewY,h);*

*p.Move(NewX,NewY);*

*end else*

*if fig='o' then*

*begin*

*NewX:=c.GetX;*

*NewY:=c.GetY;*

*Coords(a,NewX,NewY,h);*

*c.Move(NewX,NewY);*

*end else*

*if fig='e' then*

*begin*

*NewX:=e.GetX;*

*NewY:=e.GetY;*

*Coords(a,NewX,NewY,h);*

*e.Move(NewX,NewY);*

*end else*

*if fig='l' then*

*begin*

*NewX:=l.GetX;*

*NewY:=l.GetY;*

*NewX2:=l.GetX\_Line;*

*NewY2:=l.GetY\_Line;*

*Coords(a,NewX,NewY,h);*

*Coords(a,NewX2,NewY2,h);*

*l.Move(NewX,NewY,NewX2,NewY2);*

*end else*

*if fig='t' then*

*begin*

*NewX:=t.GetX;*

*NewY:=t.GetY;*

*NewX2:=t.GetX\_Line;*

*NewY2:=t.GetY\_Line;*

*NewX3:=t.GetX\_Triang;*

*NewY3:=t.GetY\_Triang;*

*Coords(a,NewX,NewY,h);*

*Coords(a,NewX2,NewY2,h);*

*Coords(a,NewX3,NewY3,h);*

*t.Move(NewX,NewY,NewX2,NewY2,NewX3,NewY3);*

*end else*

*if fig='p' then*

*begin*

*NewX:=r.GetX;*

*NewY:=r.GetY;*

*NewX2:=r.GetX\_Line;*

*NewY2:=r.GetY\_Line;*

*Coords(a,NewX,NewY,h);*

*Coords(a,NewX2,NewY2,h);*

*r.Move(NewX,NewY,NewX2,NewY2);*

*end else*

*begin*

*writeln('Error!');*

*end;*

*end;*

*if rej='s' then begin*

*if fig='po' then*

*begin*

*NewX:=10+random(600);*

*NewY:=10+random(450);*

*Coords(a,NewX,NewY,h);*

*p.Move(NewX,NewY);*

*end else*

*if fig='o' then*

*begin*

*NewX:=10+random(600);*

*NewY:=10+random(450);*

*Coords(a,NewX,NewY,h);*

*c.Move(NewX,NewY);*

*end else*

*if fig='e' then*

*begin*

*NewX:=10+random(600);*

*NewY:=10+random(450);*

*Coords(a,NewX,NewY,h);*

*e.Move(NewX,NewY);*

*end else*

*if fig='l' then*

*begin*

*NewX:=10+random(600);*

*NewY:=10+random(450);*

*NewX2:=NewX+100;*

*NewY2:=NewY+200;*

*Coords(a,NewX,NewY,h);*

*Coords(a,NewX2,NewY2,h);*

*l.Move(NewX,NewY,NewX2,NewY2);*

*end else*

*if fig='t' then*

*begin*

*NewX:=10+random(600);*

*NewY:=10+random(450);*

*NewX2:=NewX+50;*

*NewY2:=NewY+50;*

*NewX3:=NewX-50;*

*NewY3:=NewY+70;*

*Coords(a,NewX,NewY,h);*

*Coords(a,NewX2,NewY2,h);*

*Coords(a,NewX3,NewY3,h);*

*t.Move(NewX,NewY,NewX2,NewY2,NewX3,NewY3);*

*end else*

*if fig='p' then*

*begin*

*NewX:=10+random(600);*

*NewY:=10+random(450);*

*NewX2:=NewX+100;*

*NewY2:=NewY+50;*

*Coords(a,NewX,NewY,h);*

*Coords(a,NewX2,NewY2,h);*

*r.Move(NewX,NewY,NewX2,NewY2);*

*end else*

*begin*

*writeln('Error!');*

*end;*

*end;*

*until a=#27;*

*RestoreCrtMode;*

*writeln('Contine? (y/n)');*

*readln(Q);*

*end;*

*closegraph;*

*clrscr;*

*end.*

**Листинг файла FIGURA.PAS**

Unit FIGURA;

interface

TYPE

 TLocation=object

 x,y:integer;

 Constructor Init(InitX,InitY:integer);

 Function GetX: integer;

 Function GetY: integer;

 Procedure Move(NewX,NewY:integer);

 Procedure Hide;

 Procedure Draw;

 end;

 tPoint=object(tLocation)

 vh:integer;

 color:word;

 Constructor Init(InitX,InitY,InitVh:integer; col:word);

 Procedure Hide;

 Procedure Draw;

 Procedure Move(mx,my:integer);

 end;

 tLine=object(tPoint)

 Xk,Yk:integer;

 Function GetX\_Line: integer;

 Function GetY\_Line: integer;

 Constructor Init(InitX,InitY,InitVh:integer; col:word; Ixk,Iyk:integer);

 Procedure Hide;

 Procedure Draw;

 Procedure Move(mx,my,mx2,my2:integer);

 end;

 tTriang=object(tLine)

 Xk2,Yk2:integer;

 Function GetX\_Triang: integer;

 Function GetY\_Triang: integer;

 Constructor Init(InitX,InitY,InitVh:integer; col:word; Ixk,Iyk,Ixk2,Iyk2:integer);

 Procedure Hide; virtual;

 Procedure Draw; virtual;

 Procedure Move(mx,my,mx2,my2,mx3,my3:integer);

 end;

 tRectang=object(tLine)

 Constructor Init(InitX,InitY,InitVh:integer; col:word; IRxk,IRyk:integer);

 Procedure Hide;

 Procedure Draw;

 Procedure Move(mx,my,mx2,my2:integer);

 end;

 tCircle=object(tPoint)

 rad:word;

 Procedure Init(InitX,InitY,InitVh:integer; col,Irad:word);

 Procedure Hide;

 Procedure Draw;

 Procedure Move(mx,my:integer);

 end;

 tEllipse=object(tCircle)

 UgN,UgK:integer;

 Yrad:word;

 Constructor Init(InitX,InitY,InitVh:integer; col,Irad,IradY:word; IUn,IUk:integer);

 Procedure Hide;

 Procedure Draw;

 Procedure Move(mx,my:integer);

 end;

implementation

Uses Crt,Graph;

{metod tLocation}

Constructor tLocation.Init(InitX,InitY:integer);

begin

 x:=InitX; y:=InitY;

end;

Function tLocation.GetX:integer;

begin

 GetX:=X;

end;

Function tLocation.GetY:integer;

begin

 GetY:=Y;

end;

Procedure tLocation.Move(NewX,NewY:integer);

begin

 Hide;

 x:=NewX; y:=NewY;

 Draw;

end;

Procedure tLocation.Hide;

begin

end;

Procedure tLocation.Draw;

begin

end;

{metod tPoint}

Constructor tPoint.Init(InitX,InitY,InitVh:integer; col:word);

begin

 tLocation.Init(InitX,InitY);

 vh:=InitVh; color:=col;

end;

Procedure tPoint.Hide;

begin

 PutPixel(x,y,GetBkColor);

end;

Procedure tPoint.Draw;

begin

 PutPixel(x,y,color);

end;

procedure tPoint.Move(mx,my:integer);

begin

Hide;

x:=mx;

y:=my;

Draw;

end;

{metod Line}

Constructor tLine.Init(InitX,InitY,InitVh:integer; col:word; Ixk,Iyk:integer);

begin

 tPoint.Init(InitX,InitY,InitVh,col);

 Xk:=Ixk; Yk:=Iyk;

end;

Function tLine.GetX\_Line:integer;

begin

 GetX\_Line:=Xk;

end;

Function tLine.GetY\_Line:integer;

begin

 GetY\_Line:=Yk;

end;

Procedure tLine.Hide;

begin

 SetFillStyle(1,GetBkColor);

 SetColor(GetBkColor);

 Line(x,y,Xk,Yk);

end;

Procedure tLine.Draw;

begin

 SetFillStyle(1,GetBkColor);

 SetColor(color);

 Line(x,y,Xk,Yk);

end;

Procedure tLine.Move(mx,my,mx2,my2:integer);

begin

 Hide;

 x:=mx; y:=my;

 Xk:=mx2; Yk:=my2;

 Draw;

end;

{metod Triangle}

Constructor tTriang.Init(InitX,InitY,InitVh:integer; col:word; Ixk,Iyk,Ixk2,Iyk2:integer);

begin

 tLine.Init(InitX,InitY,InitVh,col,Ixk,Iyk);

 Xk2:=Ixk2; Yk2:=Iyk2;

end;

Function tTriang.GetX\_Triang:integer;

begin

 GetX\_Triang:=Xk2;

end;

Function tTriang.GetY\_Triang:integer;

begin

 GetY\_Triang:=Yk2;

end;

Procedure tTriang.Hide;

var xvs,yvs:integer;

begin

 xvs:=x; yvs:=y;

 SetFillStyle(1,GetBkColor);

 SetColor(GetBkColor);

 Line(x,y,Xk,Yk);

 Line(Xk,Yk,Xk2,Yk2);

 Line(Xk2,Yk2,xvs,yvs);

end;

Procedure tTriang.Draw;

var xvs,yvs:integer;

begin

 xvs:=x; yvs:=y;

 SetFillStyle(1,GetBkColor);

 SetColor(color);

 Line(x,y,Xk,Yk);

 Line(Xk,Yk,Xk2,Yk2);

 Line(Xk2,Yk2,xvs,yvs);

end;

Procedure tTriang.Move(mx,my,mx2,my2,mx3,my3:integer);

begin

 Hide;

 x:=mx; y:=my;

 Xk:=mx2; Yk:=my2;

 Xk2:=mx3; Yk2:=my3;

 Draw;

end;

{metod Rectagle}

Constructor tRectang.Init(InitX,InitY,InitVh:integer; col:word; IRxk,IRyk:integer);

begin

 tLine.Init(InitX,InitY,InitVh,col,IRxk,IRyk);

end;

Procedure tRectang.Hide;

begin

 SetFillStyle(1,GetBkColor);

 SetColor(GetBkColor);

 Rectangle(x,y,Xk,Yk);

end;

Procedure tRectang.Draw;

begin

 SetFillStyle(1,GetBkColor);

 SetColor(color);

 Rectangle(x,y,Xk,Yk);

end;

Procedure tRectang.Move(mx,my,mx2,my2:integer);

begin

 Hide;

 x:=mx; y:=my;

 Xk:=mx2; Yk:=my2;

 Draw;

end;

{metod Circle}

Procedure tCircle.Init(InitX,InitY,InitVh:integer; col,Irad:word);

begin

 tPoint.Init(InitX,InitY,InitVh,col);

 rad:=Irad;

end;

Procedure tCircle.Hide;

begin

 SetFillStyle(1,GetBkColor);

 SetColor(GetBkColor);

 Circle(x,y,rad);

end;

Procedure tCircle.Draw;

begin

 SetFillStyle(1,GetBkColor);

 SetColor(color);

 Circle(x,y,rad);

end;

Procedure tCircle.Move(mx,my:integer);

begin

 Hide;

 x:=mx; y:=my;

 Draw;

end;

{metod tEllipse}

Constructor tEllipse.Init(InitX,InitY,InitVh:integer;

 col,Irad,IradY:word; IUn,IUk:integer);

begin

 tCircle.Init(InitX,InitY,InitVh,col,Irad);

 Yrad:=IradY; UgN:=IUn; UgK:=IUk;

end;

Procedure tEllipse.Hide;

begin

 SetFillStyle(1,GetBkColor);

 SetColor(GetBkColor);

 Ellipse(x,y,UgN,UgK,rad,Yrad);

 Ellipse(x,y,UgN,UgK,rad,Yrad);

end;

Procedure tEllipse.Draw;

begin

 SetFillStyle(1,GetBkColor);

 SetColor(color);

 Ellipse(x,y,UgN,UgK,rad,Yrad);

end;

Procedure tEllipse.Move(mx,my:integer);

begin

 Hide;

 x:=mx; y:=my;

 Draw;

end;

end.