работа состоит из одного задания. Вид графического объекта, движение которого надо будет реализовать в работе.

**Задание:**

Написать программу, **используя объектно-ориентированный подход**, которая двигает по экрану изображение заданного графического объекта.

*Допускается: замена некоторых элементов графического объекта, изменение его цветовой гаммы.*

Реализовать два вида движения: случайное и по нажатию на клавиши со стрелками. Предусмотреть для пользователя возможность выбора одного из двух режимов движения.

Описание классов необходимо оформить в виде отдельного модуля. Иерархия классов должна включать минимум четыре класса, один из которых – абстрактный.

Работу следует выполнять в среде программирования Турбо Паскаль (версии 6.0 или 7.0) или в среде Borland Pascal.

1. **Рекомендации по выполнению курсовой работы**

Прежде, чем приступать к выполнению работы разберитесь в работе программы, приведенной в примере 2.7 конспекта лекций (§ 10). В данной программе реализована значительная часть задания работы.

Далее, проанализируйте свой графический объект. *Из каких фигур он может быть построен?* Продумайте необходимые поля и методы и опишите классы этих фигур. Создайте иерархию. Ваш заданный графический объект должен включать как минимум *(можно и больше)* 3 различные фигуры, чтобы описать минимум 3 класса *(плюс один абстрактный, итого получается 4 класса, чего и требует задание)*.

Оформите описание классов в виде модуля. Сохраните и выполните его компиляцию *(см. в лекциях “Классы и модули”).*

Теперь можете приступать к написанию самой программы. Не забудьте подключить к программе написанный Вами модуль.

Исключите из программы прямое обращение к полям.

1. **Требования к программе. Критерий оценки.**

Программа должна быть написана в соответствии с заданием своего варианта, в ней должны быть **правильно применены все изученные** нами принципы ООП, и получен правильный, соответствующий заданию результат ее работы.

В зависимости от степени выполнения изложенных требований Вам будет поставлена оценка за работу по 5-ти бальной шкале.

1. **Требования к оформлению отчета**

Необходимо представить следующее:

1. Текст задания с точным видом графического объекта, включая цвет, который реализован в программе (в файле с расширением .doc);
2. Текст программы (файл с расширением .pas) ;
3. Текст модуля (файл с расширением .pas) .

Таким образом, необходимо прислать три файла: с заданием, с программой и с модулем.

***!!! При невыполнении какого-либо из требований к оформлению работа проверяться не будет.***

**Виды графических объектов:**

****

**Пример 2.7.**

**(в котором приведены коды Программы GRAFICA и Модуля FIGURA.)**

*(Рекомендую набрать, запустить и разобрать работу данной программы)*

**Программа GRAFICA, в которой реализовано движение графических фигур**

**(сохранена в файле OOP\_DO.pas )**

**PROGRAM** GRAFICA;

**Uses** Crt,Graph,**Figura**;

**const** Path\_of\_driver='c:\tp6\bgi';

**Procedure Coords(a:char; var NewX,NewY:integer; h:integer);**

begin

 if a=#0 then begin

 a:=readkey; *{в переменную a записывается код клавиши,*

 *которую Вы нажали}*

*{#80,#75,#77,#72 – это коды клавиш со стрелками;*

 *работа оператора case: в зависимости от значения переменной a выбирается*

 *соответствующая последовательность действий; подробнее можно посмотреть в*

 *лекциях по ПЯВУ часть1}*

 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

end;

**VAR** gd,gm,NewX,NewY,h: integer;

 Q,a:char;

 str:string;

 pP:tPoint;

 pEl:tEllipse;

**BEGIN**

*{Начало работы программы. С этого момента и надо начинать разбирать*

 *работу программы}*

 gd:=Detect;

 InitGraph(gd,gm,Path\_of\_driver);

 if GraphResult<>grok then begin

 write('error:',GraphErrorMsg(GraphResult));

 halt;

 end;

 pP.Init(GetMaxX div 2,GetMaxY div 2,10,15);

 pEl.Init(GetMaxX div 2,GetMaxY div 2,10,14,70,40);

 NewX:=GetMaxX div 2; {GetMaxX - максимальное значение экрана по оси x}

 NewY:=GetMaxY div 2;

 h:=100; {влияет на скорость передвижения фигур при нажатии

 клавиш со стрелками}

 RestoreCrtMode; {восстановление текстового режима (так можно выйти из

 графического режима и вернуться к текстовому)}

 Q:='y';

 **While Q<>'n' do begin**

 write(' Выберите фигуру, для этого введите: p (если выбрали точку)');

 write (' el (если выбрали эллипс): ');

 readln(str);

*{то, что Вы наберете, т.е. p или el, запишется в строковую переменную str. Значение этой переменной нам понадобится далее, чтобы выбрать какую фигуру перемещать по экрану }*

 writeln('Нажмите клавишу Enter');

 writeln('Нажимайте клавиши со стрелками, чтобы двигать фигуру');

 readln;

 SetGraphMode(gm); {восстановление графического режима}

 **repeat**

 a:=readkey;

 Coords(a,NewX,NewY,h);

 if str='p' then pP.Move(NewX,NewY);

 if str='el' then pEl.Move(NewX,NewY);

 **until** a=#27; {#27– код клавиши Esc }

 RestoreCrtMode;

 write('Продолжить? (y/n) ');

 readln(Q); {в переменную Q запишется введенный символ: либо y, либо n}

 **end;** {окончание тела цикла While}

 closegraph;

 clrscr;

**END.**

**Модуль FIGURA, в котором реализована иерархия классов графических фигур**

**(сохранен в файле 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; virtual;

 Procedure Draw; virtual;

 end;

 **tPoint=object(tLocation)**

 vh:integer;

 color:word;

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

 Procedure Hide; virtual;

 Procedure Draw; virtual;

 end;

 **tEllipse=object(tPoint)**

 Rx,Ry:word;

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

 Procedure Hide; virtual;

 Procedure Draw; virtual;

 end;

**implementation**

**Uses** Crt,Graph;

 **{методы 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;

 **{методы 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;

 **{методы  tEllipse}**

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

 **col,InitRx,InitRy:word);**

begin

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

 Rx:=InitRx; Ry:=InitRy;

end;

**Procedure tEllipse.Hide;**

begin

 SetFillStyle(1,GetBkColor);

 SetColor(GetBkColor);

 FillEllipse(x,y,Rx,Ry);

end;

**Procedure tEllipse.Draw;**

begin

 SetFillStyle(1,GetBkColor);

 SetColor(color);

 FillEllipse(x,y,Rx,Ry);

end;

**end.**