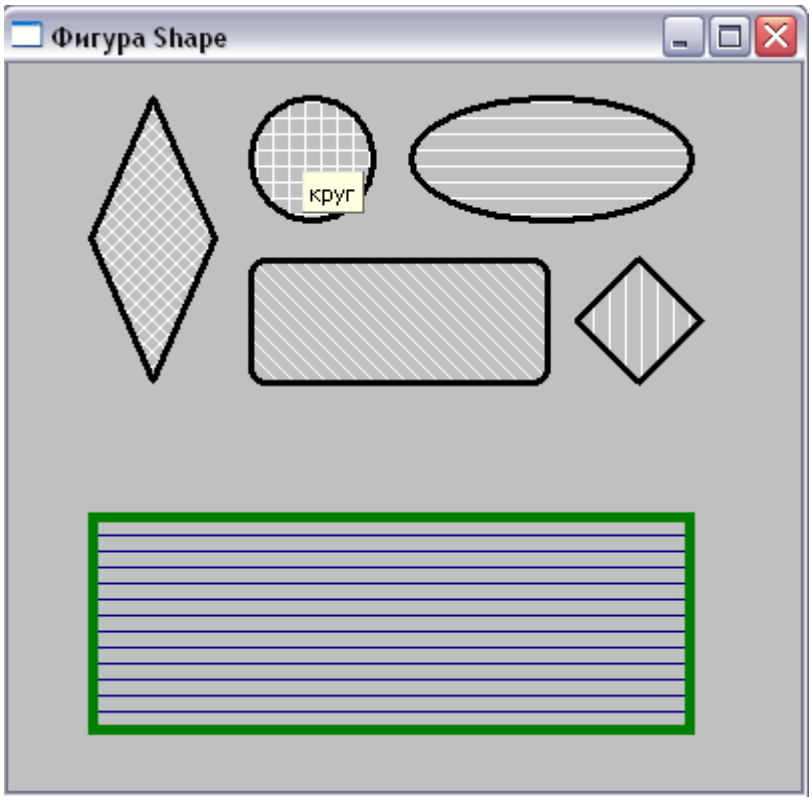
Составитель: Нуруллин А.Р., волшебник 0,5 ст.

Лабораторная работа №
Фигура Shape.

Часть 1. Фигура Shape

На форме находятся 6 фигур. В верхней части 5 и в нижней части одна, которая принимает вид той из 5 фигур в верхней части, на которую щелкнули.



Событие щелчка мыши по фигуре следует делать **OnMouseUp** или **OnMouseDown**, поскольку событие onClick для фигуры Shape отсутствует.

В событии для каждой фигуры пишем, например:

Shape1.Shape := Shape2.Shape;

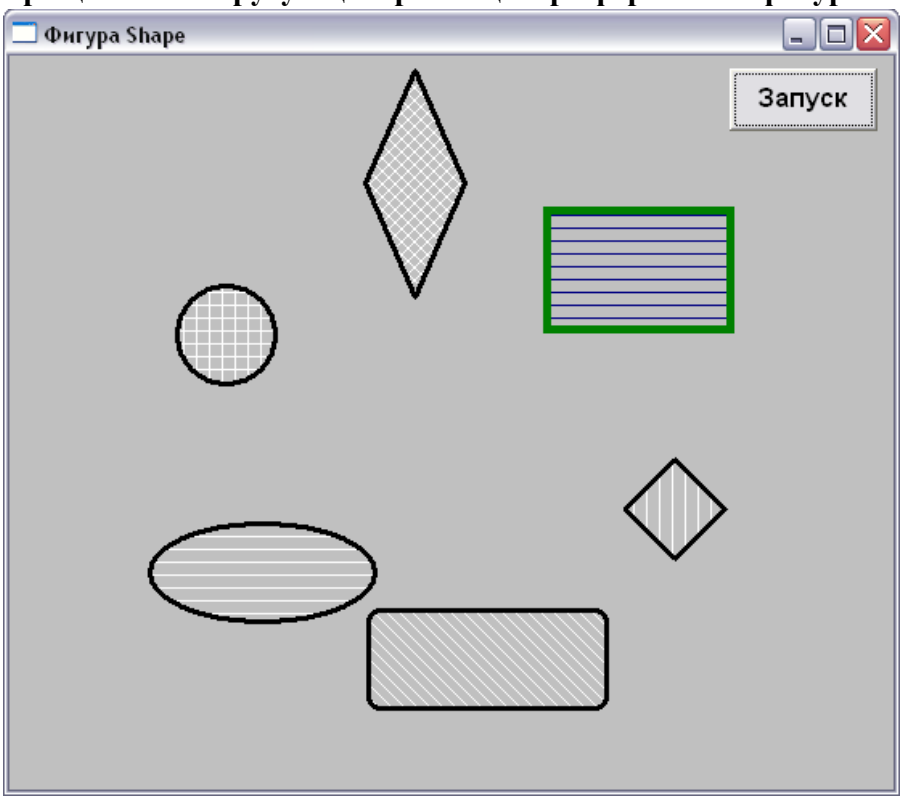
где

Shape1 – это нижняя фигура,

Shape2 – фигура по которой щелкнули

Часть 2. Массив объектов

Требуется дополнить предыдущий проект кнопкой и таймером. По нажатию на кнопку все 6 фигур начинают вращаться по кругу с центром в центре формы. По фигурам по-прежнему можно щелкать.



Очевидно, вращение фигур (высчитывание координат каждой фигуры) должно обрабатываться в процедуре работы таймера. Однако фигур много, и чтобы для каждой отдельно высчитать координаты расположения на форме понадобится много кода. Поэтому удобнее создать МАССИВ ИЗ ФИГУР. Поскольку **объекты Shape описываются типом данных TShape,** то почему бы не создать массив из этого типа данных.

Шаг 1

В области описания глобальных констант и переменных записываем:

Const n=6;

Var sh : array [1 .. n] of TShape; {массив типа TShape}

Все равно как ранее мы писали var a: array [1..10] of integer; и каждый элемент массива являлся числом типа integer;

В нашем случае мы создали массив типа **TShape** и каждый элемент массива теперь должен являться фигурой Shape, каждый элемент массива имеет свое уникальное имя.

Шаг 2

Теперь единожды заполнив массив **TShape** конкретным объектами, можно будет уже не заботиться об именах каждого объекта, а работать с ними, обращаясь к элементам массива. Заполнение массива рекомендуется делать в процедуре создания формы **FormCreate:**

```
sh[ 1 ] := Shape1;     {Первый элемент массива – фигура с именем Shape1}
sh[ 2 ] := Shape2;     {… и т.д.}
sh[ 3 ] := Shape3;
sh[ 4 ] := Shape4;
sh[ 5 ] := Shape5;
sh[ 6 ] := Shape6;
```

Шаг 3

Далее в процедуре нажатия на кнопку «ЗАПУСК» запускаем таймер.

В процедуре работы таймера, как говорилось выше, реализуем подсчет координат для каждой фигуры на круге (здесь тот же принцип, что и в проекте со звездой, имеющей **n** концов)

```
for i:= 1 to n do begin        {запускаем цикл перебора всех n фигур – элементов массива}
sh[ i ].Left :=x0 + Round(r*cos (a*pi/180) ) - (sh[ i ].Width div 2);
sh[ i ].Top :=y0 - Round(r*sin(a*pi/180) ) - (sh[ i ].Height div 2);
```

*{Подсчет координат расположения для i-той фигуры на круге, в зависимости от угла **a**, где **x0** и **y0** – это точка центра круга (центр формы), она высчитывается предварительно, также и **r** - радиус}*

```
a := a + 360 div n;     { каждая следующая фигура отстоит от предыдущей на угол 360 / n}
end;

if a>=360 then a := a - 360;     {зачем эта строка?}
```

Вопрос: Зачем при размещении очередной i-той фигуры из расчитаной координаты **x0 + Round(r*cos (a*pi/180))** мы всегда вычитаем **sh[i].Width div 2** (для **y** аналогично) ?

Процедура таймера будет выполняться через равные промежутки времени Interval, объекты будут размещаться по кругу, но не будут двигаться. Чтобы привести их в движение необходимо, чтобы с каждым новым проходом процедуры работы таймера изменялся начальный угол **a.** В нашем случае начальный угол **a** всегда равен нулю, т.е. первая фигура всегда находится в крайней правой точке окружности, остальные фигуры располагаются относительно нее.

Шаг 4

Добавьте самостоятельно 7-ую фигуру Shape, и по окружности теперь будут вращаться 7 фигур.

Шаг 5

Заставьте фигуры двигаться слева на права по форме по синусоиде, фигуры одна за другой появляются слева и исчезают за правой границей формы.

Часть 3. Новогодняя гирлянда

На форме нарисована гирлянда, состоящая из 11-ти лампочек. Каждую секунду гирлянда мигает, при этом каждая лампочка светится своим цветом. Лампочки можно разбивать, щелкая по ним мышью. Разбитые лампочки окрашены в черный цвет и больше не мигают.

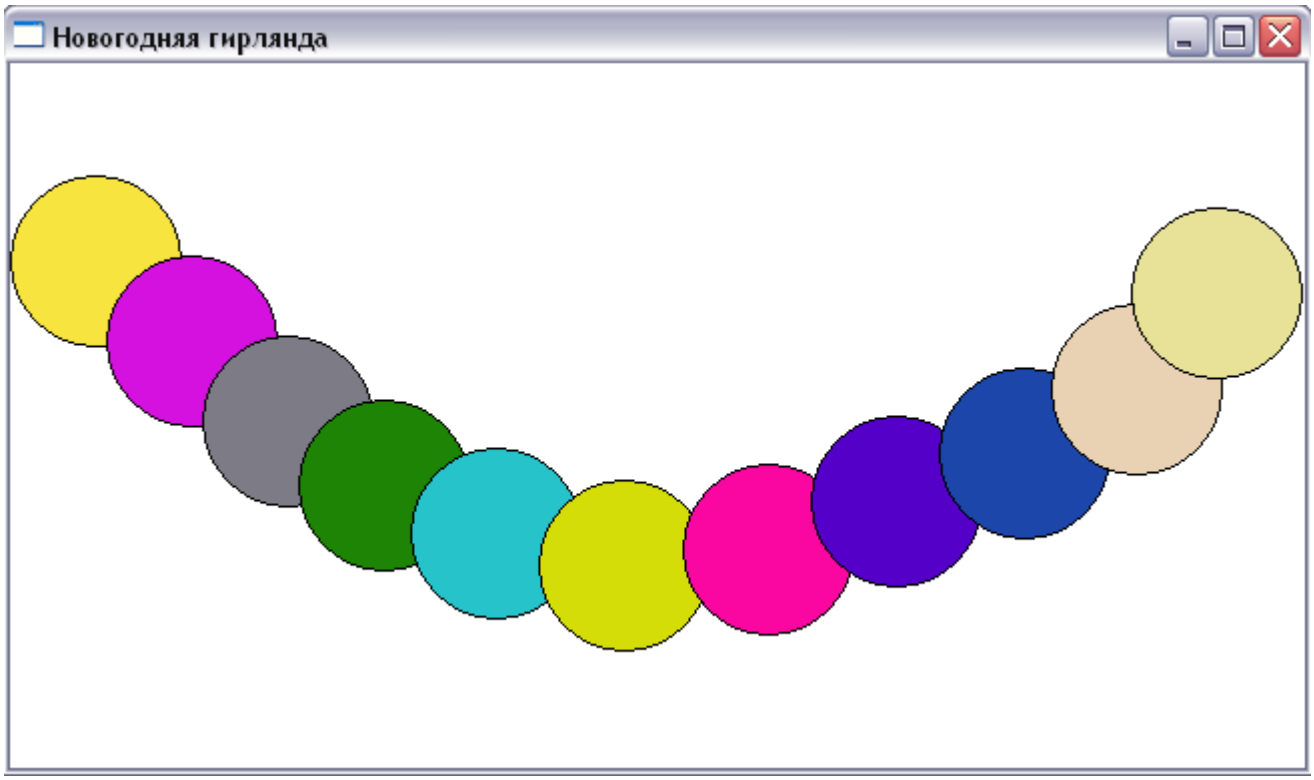
Рекомендации к выполнению:

Цвет каждой лампочки выбирается случайным образом с помощью функции

random(кол-во цветов)

Всего цветов в Lazarus - **FFFFFF**. Но это число в 16-тиричной системе счисления и чтобы указать его в функции random, необходимо предварительно перевести в 10-тичную. Т.о. для каждой лампочки будет выбираться цвет от 000000 до FFFFFF.

Для того чтобы «разбивать» лампочки понадобиться еще один массив, чтобы хранить, например, номера разбитых лампочек или индикаторы «разбитости» 0 или 1 для каждой лампочки, способов море, выбирайте сами.



Часть 4.

Пусть с течением времени гирлянда не только мигает, но и каждая лампочка меняет форму, т.е. изначально все лампочки круглые, а затем каждая становится либо квадратом, либо ромбом и т.д.

(Вводим лишнюю переменную которая опять же задается случайным образом для каждой лампочки и изменяется в диапазон от 1 до кол-ва возможных вариантов фигур. Далее удобно будет применить оператор выбора **CASE,** например ***case q of 1: sh[i].Shape:=stCircle; и т.д.*** , не забываем ставить **end** к этому оператору**)**

Часть 5.

Пусть теперь 11 шариков двигаются по синусоиде и мигают. При этом, заехав за правую границу формы каждый шарик, вновь появляется слева, т.е. двигаются по синусоиде бесконечно.

Фигура Shape

Объект предназначен для отображения на форме различных геометрических фигур.

Свойства:
Shape – задает геометрическую фигуру
Значения:
stCircle - круг
stDiamond - ромб
stEllipse - эллипс
stRectangle - прямоугольник
stRoundRect – прямоугольник с закругленным углами
stRoundSquare - квадрат
stSquaredDiamond – равноугольный ромб

Pen – карандаш (то же что и у Canvas) – границы фигуры.
Brush – кисть (то же что и у Canvas) – цвет фигуры.