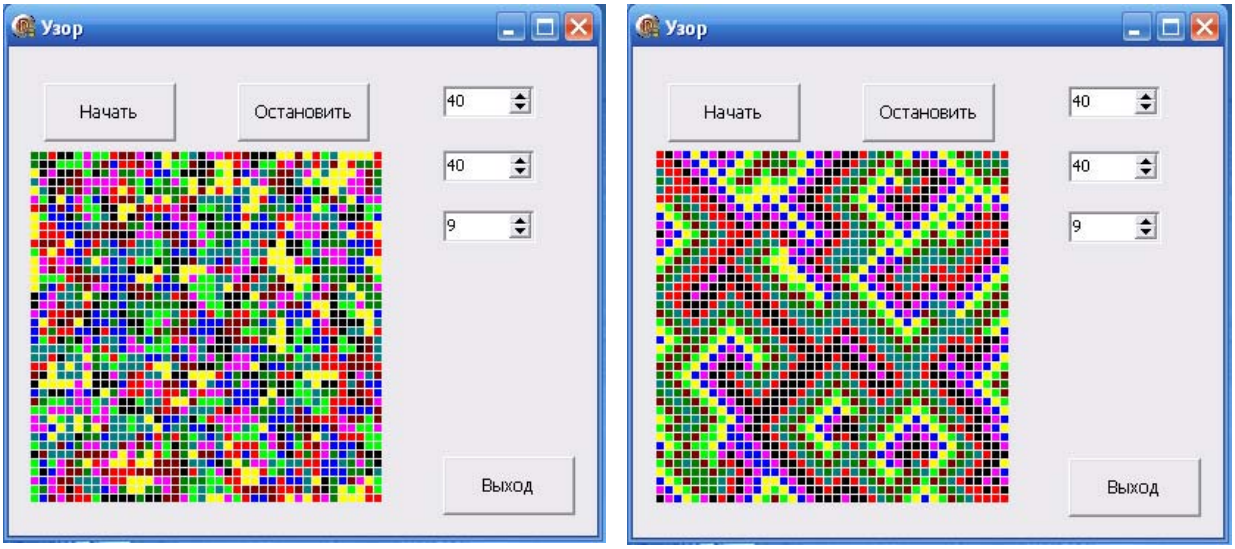
Практическая работа №12, УЗОР

Постановка задачи

Создайте программу, которая в зависимости от величин N (количество строк) и M (количества столбцов) создает разноцветный узор размером N×M, первоначально заполненяя его случайным образом различными цветами палитры (рис.1 а). Количество цветов палитры определяется величиной K, которая задается в начале программы. Рисунок узора постоянно преобразуется по определенному правилу (рис. 1 б).



а) б)

Рис.1

Правило преобразования разноцветного узора.[1]

Представим область, разбитую на клеточки. Каждая клеточка имеет свой цвет.

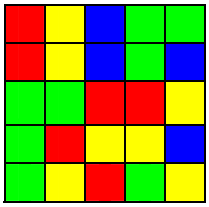
Цвета упорядочены по номерам. За красным цветом идет зеленый, за зеленым - желтый, за желтым - синий. Будем считать, что за самым последним цветом опять идет первый, то есть за синим цветом следует красный. Кроме того, будем считать, что верхний и нижний, а также правый и левый края нашей области соединены друг с другом.

Проверим для каждой клеточки цвета соседей. Если среди соседей есть клетки со «следующим» цветом, то цвет нашей клетки становится таким же. Если таких соседей нет, цвет клетки остается без изменений.

Проверка и обновление всех клеток образуют шаг изменения узора. Таких шагов может быть сколько угодно.

Для упрощения обозначим каждый цвет числом. Например, красный цвет обозначим числом 1, зеленый - 2, желтый - 3, синий - 4. В этом случае Рис.2 можно представить в виде таблицы.

[1] По материалам С.В.Симонович, Г.А.Евсеев, Занимательное программирование: Delphi - М: АСТ-ПРЕСС КНИГА, 2001



1	3	4	2	2
1	3	4	2	4
2	2	1	1	3
2	1	3	3	4
2	3	1	2	3

Рис.2

Посмотрим как изменится цвет клетки с координатами [3,2], если воспользоваться описанным ранее правилом. Наша клетка имеет цвет «2», вокруг нее расположены клетки с цветами «2», «3», «1», «1». Так как среди соседей клетки с координатами [3,2] есть клетки со «следующим» цветом («3»), то цвет нашей клетки становится «3».

1	3	4	2	2
1	3	4	2	4
2	**2**	1	1	3
2	1	3	3	4
2	3	1	2	3

1	3	4	2	2
1	3	4	2	4
2	**3**	1	1	3
2	1	3	3	4
2	3	1	2	3

В программе цветов может быть не четыре, а намного больше. На первый взгляд может показаться, что, каким бы ни был начальный узор, в конце концов получится одноцветная область. Но это мнение ошибочно. Что получитсч на самом деле, будет видно, когда программа заработает.

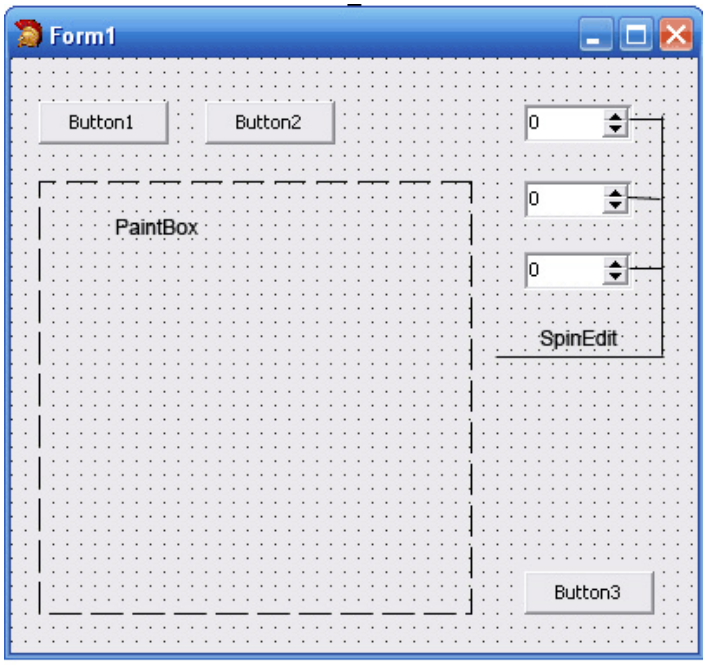


Рис.3

Нам понадобятся следующие компоненты:

№ п/п	Наименование компонента	Страница	Для чего предназначен
1	Button1	Standard	Кнопка начала построения узора
2	Button2	Standard	Кнопка остановки построения узора или продолжения построения узора
3	Button3	Standard	Кнопка выхода из программы
4	SpinEdit1	Samples	Количество строк в узоре

5	SpinEdit2	Samples	Количество столбцов в узоре
6	SpinEdit3	Samples	Количество цветов в узоре
7	PaintBox	System	Поле построения узора

Новым в этой работе являются:

- организация бесконечного цикла

Основные этапы программы

1. Задаются начальные значения количество строк (N_SIZE) и столбцов (M_SIZE) в рисунке, количество цветов (K_SIZE) в палитре.

2. Формируется исходный числовой массив (AR), состоящий из цветов заданной палитры. Для заполнения массива используется функция Random.

3. Расчитывается размер цветных прямоугольников, которыми будет заполняться поле PaintBox при формировании рисунка, в зависимости от велечин количества строк (N_SIZE) и столбцов (M_SIZE).

4. Заполняется поле PaintBox разноцветными прямоугольниками. Значение цвета прямоугольника определяется на основании числового массива AR.

5. После нажатия кнопки «Начать» программа приступает к преобразованию узора:

- последовательно просматривает все значения массива AR и по заданному правилу формирует новый массив NEW_AR;
- переписывает массив NEW_AR в массив AR;
- формирует новое экранное изображение в зависимости от значений элементов массива AR.

6. П.5 повторяется до тех пор пока не будет нажата кнопка «Остановить», которая останавливает процесс формирования рисунка и меняет при этом название кнопки «Остановить» на «Продолжить».

7. Для продолжения формирования рисунка необходимо нажать кнопку «Продолжить», которая при этом изменяет свое название на «Остановить».

8. Можно изменить начальные значения количество строк (N_SIZE) и столбцов (M_SIZE) в рисунке, количество цветов (K_SIZE) в палитре, для этого необходимо сначала нажать кнопку «Остановить», а затем поменять начальные значения и нажать кнопку «Начать».

9. Выйти из программы можно, если сначала остановить процесс формирования узора (нажать кнопку «Остановить»), а затем нажать кнопку «Выход».

Немного теории

Построение графических изображений осуществляется на поверхность объекта (формы или др.компонент). У ряда объектов из библиотеки визуальных компонент есть свойство **Canvas**. Для того чтобы вывести на поверхность объекта графический элемент (прямую линию, окружность, прямоугольник и т. д.), необходимо применить к свойству **Canvas** этого объекта соответствующий метод.

Canvas является в свою очередь объектом, объединяющим в себе поле для рисования, карандаш (**Pen**), кисть (**Brush**) и шрифт (**Font**). **Canvas** обладает также рядом графических методов: **LineTo**, **Ellipse**, **Rectangle**, **TextOut**, **Arc**, и др. Эти методы обеспечивают вывод графических примитивов (линий, окружностей, прямоугольников, текстов и т. д.), а свойства позволяют задать характеристики выводимых графических примитивов: цвет, толщину и стиль линий; цвет и вид заполнения областей; характеристики шрифта при

выводе текстовой информации.

Методы вывода графических примитивов рассматривают свойство **Canvas** как некоторый абстрактный холст, на котором они могут рисовать (*canvas* переводится как «поверхность», «холст для рисования»).

Холст состоит из отдельных точек – пикселей. Положение пикселя характеризуется его горизонтальной (X) и вертикальной (Y) координатами. Левый верхний пиксель имеет координаты (0, 0). Координаты возрастают сверху вниз и слева направо. Значения координат правой нижней точки холста зависят от размера холста.

Метод **Rectangle** рисует прямоугольник с одним углом в точке, указанной параметрами x1, y1, и противоположным диагональным углом в точке, заданной параметрами x2,y2. Рамка прямоугольника рисуется текущим пером (**Pen**) и заполняется текущим фоном (**Brush**).

Объект.Canvas. Rectangle (x1,y1, x2,y2)

где: объект – имя объекта (компонента), на поверхности которого выполняется вычерчивание.

Свойство **Canvas** доступно только во время работы приложения.

Canvas.Brush – свойство фона.

Свойство **Canvas.Brush.Color** определяет необходимый цвет

Инструкция **Canvas.FillRect(ClientRect)** очищает всю рабочую область компонента, заливая ее установленным ранее цветом.

Свойство **Canvas.Brush.Bitmap** устанавливает рисунок в качестве фона – присвоить переменную (имя файла) с растровым рисунком.

Canvas.Pen – свойства пера.

Canvas.MoveTo(x,y) – устанавливает перо в заданную точку, где x и y - координаты точки, относительно компонента. После этой команды перо установлено, но точка не нарисована.

Canvas.LineTo(x,y) – провести линию от текущего положения пера до заданной точки (x,y).

Canvas.Pixels[x,y]:=ЦВЕТ_ТОЧКИ – поставить на холсте точку с координатами (x,y) определенного цвета.

Canvas.Pen.Width:=ТОЛЩИНА_В_ТОЧКАХ.

Canvas.Pen.Color:=ЦВЕТ.

План разработки программы

1. Откройте новый проект.

2. Разместите в блоке реализации после слова **implementation** описание констант и переменных:

```
Const N_MAX=100; //Максимальное количество строк
      M_MAX=100; //Максимальное количество столбцов
      K_MAX=20;  //Максимальное количество цветов палитры
      RAZX=350;   //Размер поля рисунка по оси X
      RAZY=350;   //Размер поля рисунка по оси Y
      COLORS:array[1..20]of tcolor= //Массив палитры цветов
             (clblack,clmaroon,clgreen,clolive,clnavy,
              clpurple,clteal,clgray,clsilver,clred,
              cllime,clyellow,clblue,clfuchsia,claqua,
              clwhite,clmoneygreen,clskyblue,clcream,clmedgray);
Type AR_TYPE= Array[1..RAZX,1..RAZY] of Integer;
Var N_SIZE : Integer;  // Количество строк узора
    M_SIZE : Integer;  // Количество столбцов узора
    K_SIZE : Integer;  // Количество цветов в узоре
    C_X : Integer;     // Размер прямоугольника (клетки)по X
    C_Y : Integer;     // Размер прямоугольника (клетки)по Y
    FLAG: Boolean;  // Флаг останова или продолжения построения
    AR,NEW_AR:AR_TYPE;
      // Массивы цветов до и после преобразования
```

3. Разместите в форме компоненты в соответствии с Рис.3. Для этого выделите объект **Form1**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **OnCreat**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий текст:

```
// Формирование элементов формы
Form1.Caption:='Узор';

Button1.Font.Size:=9;
Button1.Caption:='Начать';

Button2.Font.Size:=9;
Button2.Caption:='Остановить';
Button2.Enabled:=False;   //Кнопка недоступна

Button3.Font.Size:=9;
Button3.Caption:='Выход';

SpinEdit1.MinValue:=3;    //Минимальное количество строк
SpinEdit1.MaxValue:=N_MAX;  //Максимальное количество
SpinEdit1.Value:=10;      //Текущее значение

SpinEdit2.MinValue:=3;    //Минимальное количество столбцов
SpinEdit2.MaxValue:=M_MAX;  //Максимальное количество
SpinEdit2.Value:=10;      //Текущее значение

SpinEdit3.MinValue:=2;    //Минимальное количество
SpinEdit3.MaxValue:= K_MAX;  //Максимальное количество
```

```
SpinEdit3.Value:=3;        //Текущее значение

PaintBox1.Width:=RAZX;     //Определение размера поля PaintBox
PaintBox1.Height:=RAZY;
FLAG:= True;
```

4. Выделите объект **Button1**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **OnClick**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий текст:

```
procedure TForm1.Button1Click(Sender: TObject);
var i,j:integer;
begin

 // Определение текущих значений
 N_SIZE:=SpinEdit1.Value;
 M_SIZE:=SpinEdit2.Value;
 K_SIZE:=SpinEdit3.Value;

 // Установка недоступности компонент
 SpinEdit1.Enabled:=False;
 SpinEdit2.Enabled:=False;
 SpinEdit3.Enabled:=False;

 Button2.Enabled:=True;   //Кнопка доступна

 // Заполнение элементов массива
 Randomize;
 for i:=1 to N_SIZE do
   for j:=1 to M_SIZE do
     AR[i,j]:= 1 + Random(K_SIZE);

 // Очистка поля для построения узора
 Form1.PaintBox1.Canvas.Pen.Color:=clBtnFace;
 Form1.PaintBox1.Canvas.Brush.Color:= clBtnFace;
 Form1.PaintBox1.Canvas.Rectangle(0,0,RAZX,RAZY);

 // Определение размера одной клетки узора
 C_X:=RAZX div N_SIZE;
 C_Y:=RAZY div M_SIZE;

 //Процедура первоначального формирования узора
 IMAG(AR,N_SIZE,M_SIZE);

 FLAG:=True;
 //Процедура формирования узора
 USOR;
end;
```

5. Разместите после блока описания переменных (см.п.2) две процедуры, которые были использованы в предыдущем п.4.

Процедура построения изображения узора

```
procedure IMAG(AR:AR_TYPE;N,M:Integer);
var I,J:Integer;
begin
// Построение узора
```

```
 for i:=1 to N do
  for j:=1 to M do
   begin
    Form1.PaintBox1.Canvas.Pen.Color:= Colors[AR[i,j]];
    Form1.PaintBox1.Canvas.Brush.Color:= Colors[AR[i,j]];
Form1.PaintBox1.Canvas.Rectangle(C_X*(i-1),C_Y*(j-1),C_X*i-1,C_Y*j-1);
   end;
end;
```

Пояснение

В процедуре организованы два цикла - по количеству строк и по количеству столбцов для перебора всех элементов массива AR. Для каждого элемента определяется цвет выводимого прямоугольника и его координаты.

Процедура обработки узора

Сначала приведем общий вид данной процедуры и переменные, которые будут использованы.

```
Procedure USOR;
Var C, // Новый цвет текущей клетки
    U, // Индекс клетки сверху
    D, // Индекс клетки снизу
    R, // Индекс клетки справа
    L  // Индекс клетки слева
       :Integer;
    i,j: Integer;      // Переменные цикла
begin
 While FLAG do
 // Бесконечный цикл, пока флаг остановки будет равен TRUE
 begin
  //Блок обработки массива AR
  ...

 IMAG(AR,N_SIZE,M_SIZE);
 Sleep(100);
 Application.ProcessMessages; //Обработка всей очереди сообщений
 end;
end;
```

Пояснение

В процедуре используется бесконечный цикл **While FLAG do**, т.к. переначально переменная **FLAG** равна **True**. Этот цикл сначала обрабатывает все элементы массива AR, а затем выполняет формирование нового изображения (процудура **IMAGE**). В конце цикла записаны две процедуры:

Sleep(100);
Application.ProcessMessages;

Примечание

- Процедура **Sleep** выполняет остановку программы на заданное время в милисекундах для того, чтобы мы смогли зафиксировать свой взгляд на изображении. **Sleep(100)** – задержка на 0,1 сек.
- Процедура **Application.ProcessMessages** заставляет программу взять все посланные приложению сообщения, которые находятся на данный момент в

очереди сообщений, и обработать их. Это нам необходимо для того, чтобы можно было бы остановить работу программу.

Блок обработки элементов массива AR

```
for i:=1 to N_SIZE do
   for j:=1 to M_SIZE do
     begin
      // Определение нового цвета клетки
           ...
     end;
 //Перезапись массива NEW_AR в AR
 for i:=1 to N_SIZE do
    for j:=1 to M_SIZE do
       AR[i,j]:= NEW_AR[i,j];
 IMAG(AR,N_SIZE,M_SIZE);
```

Пояснение

В начале записаны два вложенных цикла, которые позволяют просмотреть все значения элементов массива **AR** и сформировать новый массив **NEW_AR** с измененными значениями. Затем переписываем элементы нового массива **NEW_AR** в старый массив **AR** и передаем управление процедуре формирования нового узора на экране.

Текст блока определения нового цвета клетки

```
 C:=AR[i,j]+1; // Вычисление следующего цвета клетки
 if C>K_SIZE then C:=1; // После последнего цвета идет первый
 // Вычисление индексов клеток
 U:=i-1; // сверху
 if U=0 then U:=N_SIZE;
 D:=i+1; // снизу
 if D>N_SIZE then D:=1;
 L:=j-1; // слева
 if L=0 then L:=M_SIZE;
 R:=j+1; // справа
 if R>M_SIZE then R:=1;

 // Если среди соседей есть «следующий» цвет, то новая клетка
 // приобретает этот цвет, в противном случае она не изменяется
 if (AR[U,j]=C)or (AR[D,j]=C)or AR[i,L]=C)or (AR[i,R]=C)
     then  NEW_AR[i,j]:=C
     else  NEW_AR[i,j]:=AR[i,j];
```

6. Выделите объект **Button2**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **OnClick**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий текст:

```
If FLAG=True
 Then
   begin
    FLAG:=False;
    Button2.Caption:='Продолжить';
   // Установка доступности компонент
    SpinEdit1.Enabled:=True;
    SpinEdit2.Enabled:=True;
    SpinEdit3.Enabled:=True;
   end
```

```
  Else
   begin
    FLAG:=True;
    Button2.Caption:='Остановить';
    // Установка недоступности компонент
    SpinEdit1.Enabled:=False;
    SpinEdit2.Enabled:=False;
    SpinEdit3.Enabled:=False;
    USOR;
   end;
```

7. Самостоятельно добавьте обработку кнопки «Выход».

8. Сохраните проект окончательно, запустите и протестируйте его.

Задание для самостоятельного выполнения

1. Если вы внимательно протестировали программу, то обратили внимание на не соответствие свойств Caption кнопок Button1 и Button2 после следующей последовательности нажатия кнопок: Button1 («Начать») - Button2 («Остановить») - Button1 («Начать»). Самостоятельно исправьте эту ошибку.

2. Напишите программу, которая будет выводить на экран цветные окружности, генерируя случайным образом координаты, радиус и цвет.

3. Внесите изменения в программу, чтобы на экран выводилось 10 окружностей, затем первая окружность стиралась и выводилась на экран следующая, которая становится 10-ой. Этот процесс повторяется до нажатия любой клавиши.

Листинг программы

```
unit Unit12;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Spin, ExtCtrls;

type
  TForm1 = class(TForm)
    PaintBox1: TPaintBox;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    SpinEdit3: TSpinEdit;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}
 Const N_MAX=50;
       M_MAX=50;
       K_MAX=20;
       RAZX=250;     // Размер поля узора по X
       RAZY=250;     // Размер поля узора по Y
      COLORS:array[1..20]of tcolor= //Массив цветов
(cllime,clyellow,clblue,clfuchsia,clblack,clred,clteal,
clgreen,clmaroon,clolive,clnavy,
clpurple,clgray,clsilver,claqua,clwhite,
clmoneygreen,clskyblue,clcream,clmedgray);
Type AR_TYPE= Array[1..RAZX,1..RAZY] of Integer;
Var N_SIZE : Integer;  // Текущее количество строк узора
    M_SIZE : Integer;  // Текущее количество столбцов узора
    K_SIZE : Integer;  // Текущее количество цветов в узоре
    C_X : Integer;     // Размер прямоугольника (клетки) по X
    C_Y : Integer;     // Размер прямоугольника (клетки) по Y
    AR,NEW_AR:AR_TYPE;
    FLAG : Boolean; // Флаг останова или запуска узора
procedure IMAG(AR:AR_TYPE;N,M:Integer);
// Процедура построения изображения узора
var I,J:Integer;
begin
```

```
// Построение узора
  for i:=1 to N do
    for j:=1 to M do
      begin
        Form1.PaintBox1.Canvas.Pen.Color:= Colors[AR[i,j]];
        Form1.PaintBox1.Canvas.Brush.Color:= Colors[AR[i,j]];
        Form1.PaintBox1.Canvas.Rectangle(C_X*(i-1),C_Y*(j-1),C_X*i-
1,C_Y*j-1);
      end;
end;
Procedure USOR;
Var C, // Цвет текущей клетки
    U, // Цвет клетки сверху
    D, // Цвет клетки снизу
    R, // Цвет клетки справа
    L  // Цвет клетки слева
       :Integer;
    i,j: Integer;      // Переменные цикла
begin
While FLAG do // бесконечный цикл, пока флаг остановки будет равен
TRUE
   begin
   //Блок обработки массива AR
   for i:=1 to N_SIZE do
   for j:=1 to M_SIZE do
     begin
       C:=AR[i,j]+1;
       if C>K_SIZE then C:=1;
       U:=i-1;
       if U=0 then U:=N_SIZE;
       D:=i+1;
       if D>N_SIZE then D:=1;
       L:=j-1;
       if L=0 then L:=M_SIZE;
       R:=j+1;
       if R>M_SIZE then R:=1;

       if (AR[U,j]=C) or (AR[D,j]=C) or
          (AR[i,L]=C) or (AR[i,R]=C)
          then  NEW_AR[i,j]:=C
          else  NEW_AR[i,j]:=AR[i,j];
     end;
  //Перезапись массива NEW_AR в AR
  for i:=1 to N_SIZE do
    for j:=1 to M_SIZE do
           AR[i,j]:= NEW_AR[i,j];
  IMAG(AR,N_SIZE,M_SIZE);
  Sleep(100);
  Application.ProcessMessages; // Обработка всей очереди сообщений
 end;
end;
procedure TForm1.Button1Click(Sender: TObject);
var i,j:integer;

begin

 // Определение текущих значений
 N_SIZE:=SpinEdit1.Value;
 M_SIZE:=SpinEdit2.Value;
```

```
 K_SIZE:=SpinEdit3.Value;

 // Установка недоступности компонент
 SpinEdit1.Enabled:=False;
 SpinEdit2.Enabled:=False;
 SpinEdit3.Enabled:=False;

 Button2.Enabled:=True;   //Кнопка доступна

 // Заполнение элементов массива
 Randomize;
 for i:=1 to N_SIZE do
   for j:=1 to M_SIZE do
     AR[i,j]:= 1 + Random(K_SIZE);

 // Очистка поля для построения узора
 Form1.PaintBox1.Canvas.Pen.Color:=clBtnFace;
 Form1.PaintBox1.Canvas.Brush.Color:= clBtnFace;
 Form1.PaintBox1.Canvas.Rectangle(0,0,RAZX,RAZY);
 // Определение размера одной клетки узора
C_X:=RAZX div N_SIZE;
C_Y:=RAZY div M_SIZE;
 //Процедура первоначального формирования узора
 IMAG(AR,N_SIZE,M_SIZE);

 FLAG:=True;
 //Процедура формирования узора
 USOR;

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
If FLAG=True Then
   begin
    FLAG:=False;
    Button2.Caption:='Продолжить';
    // Установка доступности компонент
    SpinEdit1.Enabled:=True;
    SpinEdit2.Enabled:=True;
    SpinEdit3.Enabled:=True;
   end
  Else
   begin
    Button2.Caption:='Остановить';
    // Установка недоступности компонент
    SpinEdit1.Enabled:=False;
    SpinEdit2.Enabled:=False;
    SpinEdit3.Enabled:=False;
    FLAG:=True; USOR;
   end;
 end;
procedure TForm1.Button3Click(Sender: TObject);
begin
Close;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
```

```
// Определение начальных значений
N_SIZE:=20;
M_SIZE:=20;
K_SIZE:=6;

// Формирование элементов формы

Form1.Caption:='Узор';

Button1.Font.Size:=9;
Button1.Caption:='Начать';

Button2.Font.Size:=9;
Button2.Caption:='Остановить';
Button2.Enabled:=False;   //Кнопка недоступна

Button3.Font.Size:=9;
Button3.Caption:='Выход';

SpinEdit1.MinValue:=3;      //Минимальное количество
SpinEdit1.MaxValue:=N_MAX;  //Максимальное количество
SpinEdit1.Value:=N_SIZE;    //Текущее значение

SpinEdit2.MinValue:=3;     //Минимальное количество
SpinEdit2.MaxValue:=M_MAX; //Максимальное количество
SpinEdit2.Value:=M_SIZE;   //Текущее значение

SpinEdit3.MinValue:=2;     //Минимальное количество
SpinEdit3.MaxValue:=K_MAX; //Максимальное количество
SpinEdit3.Value:=K_SIZE;   //Текущее значение

PaintBox1.Width:=RAZX;  //Определение размера поля PaintBox
PaintBox1.Height:=RAZY;

FLAG:= False;
end;

end.
```