Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Самарский государственный технический университет»

Факультет автоматики информационных технологий
Кафедра «Автоматика и управление в технических системах»

Курсовая работа

по дисциплине «Программирование и основы алгоритмизации»

Вариант № 2(5*)

Выполнил: студент:
1-АИТ-1 Чекушкин П.М.
Проверил:
к.т.н., доцент Мандра А.Г.

Самара 2017

1 Задание

Обработка изображения на основе разности яркости окрестности, оставить только те точки, у которых двоичный код разности с двумя переходами 0-1, 1-0. Размер точки задается при запуске: 1px, 2px, 3px и тд. Порог для определения разности задается при запуске. Исходное изображение – цветное. В результате наложить результирующее изображение на исходное.

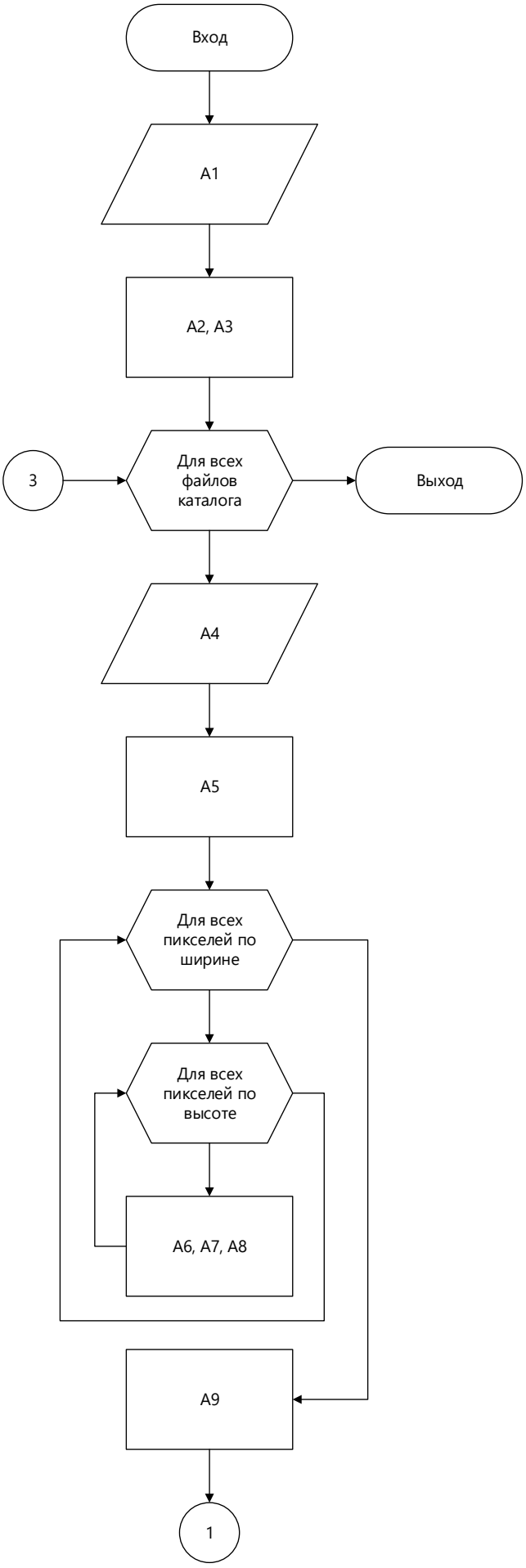
2 Теоретические сведения

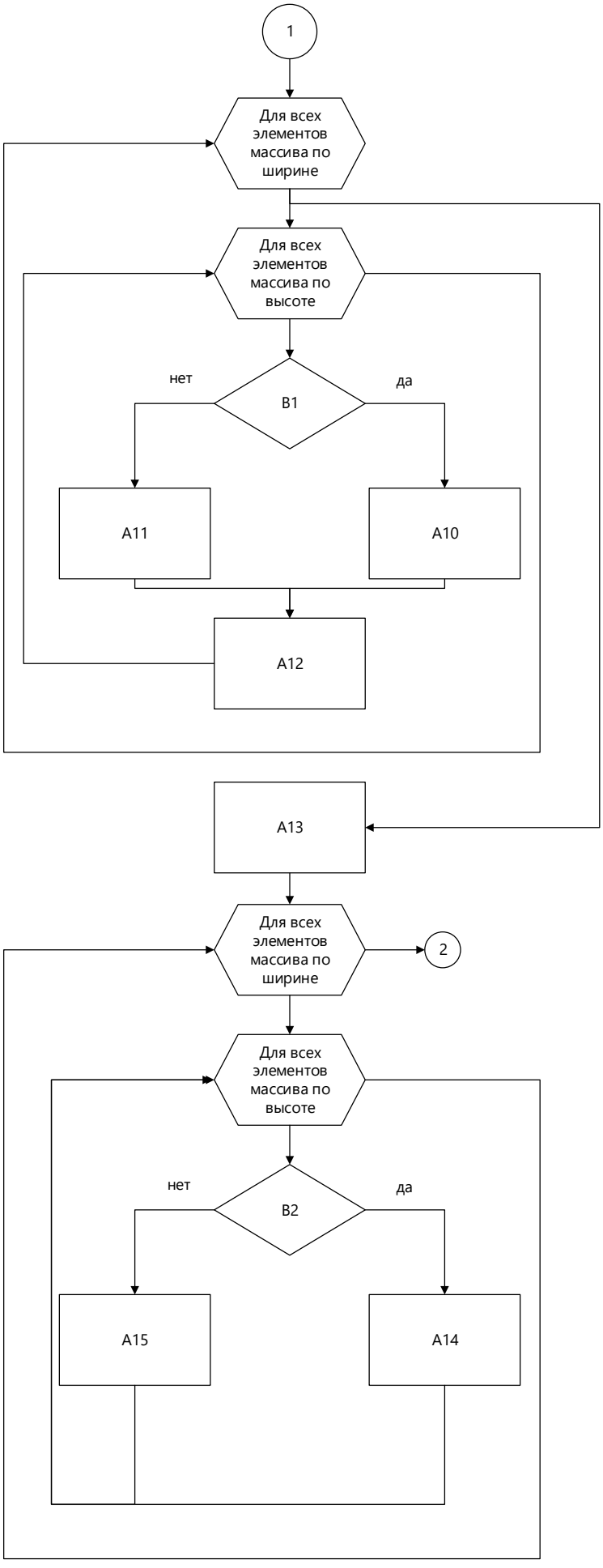
Программа работает с изображениями в формате ‘jpg’. Исходные изображения находятся в папке ‘in’, результирующие изображения находятся в папке ‘out’. Программа сохраняет изображения в папку ‘out’, если таковая имеется, если папка отсутствует, то она будет создана. Запрашиваем у пользователя параметры по которым будет происходить обработка изображения. С каждым изображением работаем по порядку. После считывания изображения создаем двумерный массив в который потом запишем среднюю яркость пикселей. Проходим по каждому пикселю изображения, получаем значение RGB и вычисляем яркость пикселя. Записываем полученную яркость в массив, созданный ранее. Создаем двумерный массив под маску пикселей, которую получим на следующем шаге. Проходим по массиву яркости пикселей, вычисляем среднюю яркость для текущего пикселя и среднюю яркость пикселей, которые его окружают. Средняя яркость рассчитывается на основе размера точки заданной пользователем. Сравниваем среднюю яркость текущего пикселя и среднюю яркость пикселей его окружающих с порогом разности яркости пикселей заданным пользователем. После сравнения создаем маску для каждого пикселя и записываем в массив. Создаем двумерный массив соответствия маски условию, значение которого получим на следующем шаге. Проходим по массиву маски пикселей. Проверяем текущую маску на соответствие условию. Записываем в массив положительный или отрицательный результат. Проходим по массиву соответствия маски условию и заменяем пиксели в изображении, если это потребуется. Сохраняем полученное изображение.

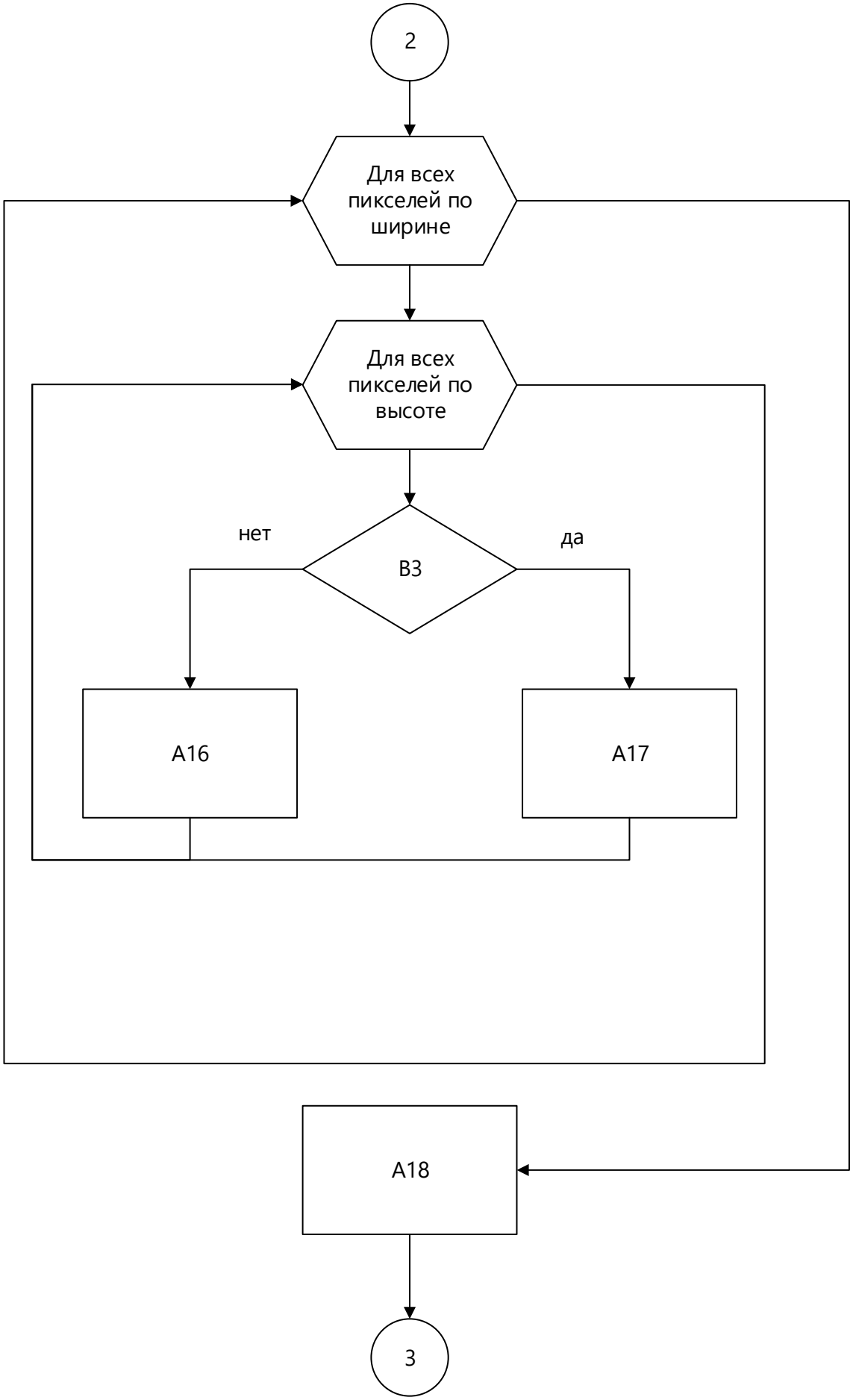
3 Схема алгоритма

Код действия	***Наименование действия***
A1	Опросить пользователя
A2	Создать каталог для результирующих изображений
A3	Получить список всех файлов заданного каталога
A4	Прочитать изображение
A5	Создать массив для хранения разности яркости пикселей
A6	Получить значение RGB по X и Y для текущего пикселя
A7	Посчитать значение яркости для текущего пикселя
A8	Полученное значение записать в массив
A9	Создать массив для хранения маски яркости пикселя
A10	В конец маски записать 1
A11	В конец маски записать 0
A12	Полученное значение записать в массив
A13	Создаем массив соответствия маски условию
A14	Записать в массив 1
A15	Записать в массив 0
A16	Оставляем текущий пиксель
A17	Заменяем текущий пиксель в изображении
A18	Сохранить результирующее изображение в файл

Код условия	***Наименование условия***
B1	Разность яркости пикселей по модулю больше порога разности яркости пикселей заданным пользователем
B2	Количество переходов в маске удовлетворяет условию
B3	Замена пикселя удовлетворяет условию


Вход
A1
A2, A3
3
Для всех файлов каталога
Выход
A4
A5
Для всех пикселей по ширине
Для всех пикселей по высоте
A6, A7, A8
A9
1


1
Для всех элементов массива по ширине
Для всех элементов массива по высоте
нет
B1
да
A11
A10
A12
A13
Для всех элементов массива по ширине
2
Для всех элементов массива по высоте
нет
B2
да
A15
A14


2
Для всех пикселей по ширине
Для всех пикселей по высоте
нет
B3
да
A16
A17
A18
3

4 Листинг программы

```
uses
  graphABC,System.IO;
var
  image, image1: picture;
  st, fileName: string;
  pointSize: integer;
  differenceIntensity: integer;
  massY: array [,] of real;
  Y: real;
  i, j, z, c, v: integer;
  massMask : array[,] of string;
  mask : string;
  massBoolean : array[,] of integer;
  count : integer;
  str : string;
  bol : boolean;
  k, k1, k2, k3,k4, k5, k6, k7, k8, p, p1, p2, p3, p4, p5, p6, p7, p8 :
real;
begin
  writeln('Введите размер точки в px: ');
  readln(pointSize);
  writeln('Введите порог для определения разности: ');
  readln(differenceIntensity);
  MkDir(GetCurrentDir+'\out');
  var files := Directory.GetFiles(GetCurrentDir+'\in',
'*.jpg',SearchOption.AllDirectories);
  for var ff:=0 to files.Length-1 do
    begin
      st := files[ff];
      fileName := ExtractFileName(st);
      image := Picture.Create(st);

      SetLength(massY, image.Width, image.Height);
      for i := 0 to image.Width - 1 do
        begin
          for j := 0 to image.Height - 1 do
            begin
              Y := 0.299 * image.GetPixel(i, j).R + 0.587 *
image.GetPixel(i, j).G + 0.144 * image.GetPixel(i, j).B;
              massY[i, j] := Y;
            end;
        end;

      mask := '';
      SetLength(massMask, image.Width, image.Height);
        for i := pointSize to massY.GetLength(0) - 2 * pointSize do
          begin
            for j :=pointSize to massY.GetLength(1) - 2 * pointSize do
              begin

                for c := i to i + pointSize - 1 do
                  begin
                    for v := j to j + pointSize - 1 do
                      begin
```

```
        k := massY[c,v] + k;
      end;
  end;
p := k / pointSize * pointSize;

for c := i-1 downto i - pointSize do
  begin
    for v := j-1 downto j - pointSize do
      begin
        k1 := massY[c,v] + k1;
      end;
  end;
p1 := k1 / pointSize * pointSize;

for c := i to i + pointSize - 1 do
  begin
    for v := j-1 downto j - pointSize do
      begin
        k2 := massY[c,v] + k2;
      end;
  end;
p2 := k2 / pointSize * pointSize;

for c := i + pointSize to i + pointSize*2 - 1 do
  begin
    for v := j-1 downto j - pointSize do
      begin
        k3 := massY[c,v] + k3;
      end;
  end;
p3 := k3 / pointSize * pointSize;

for c := i-1 downto i - pointSize do
  begin
    for v := j to j + pointSize - 1 do
      begin
        k4 := massY[c,v] + k4;
      end;
  end;
p4 := k4 / pointSize * pointSize;

for c := i + pointSize to i + pointSize*2 - 1 do
  begin
    for v := j to j + pointSize - 1 do
      begin
        k5 := massY[c,v] + k5;
      end;
  end;
p5 := k5 / pointSize * pointSize;

for c := i-1 downto i - pointSize do
  begin
    for v := j + pointSize  to j + pointSize*2 - 1 do
      begin
        k6 := massY[c,v] + k6;
      end;
  end;
p6 := k6 / pointSize * pointSize;
```

```
        for c := i to i + pointSize - 1 do
          begin
            for v := j + pointSize  to j + pointSize*2 - 1 do
              begin
                k7 := massY[c,v] + k7;
              end;
          end;
        p7 := k7 / pointSize * pointSize;

        for c := i + pointSize to i + pointSize*2 - 1 do
          begin
            for v := j + pointSize  to j + pointSize*2 - 1 do
              begin
                k8 := massY[c,v] + k8;
              end;
          end;
        p8 := k8 / pointSize * pointSize;

        if abs(p-p1)>differenceIntensity then  mask := mask + '0'
        else mask := mask + '1';
        if abs(p-p2)>differenceIntensity then  mask := mask + '0'
        else mask := mask + '1';
        if abs(p-p3)>differenceIntensity then  mask := mask + '0'
        else mask := mask + '1';
        if abs(p-p4)>differenceIntensity then  mask := mask + '0'
        else mask := mask + '1';
        if abs(p-p5)>differenceIntensity then  mask := mask + '0'
        else mask := mask + '1';
        if abs(p-p6)>differenceIntensity then  mask := mask + '0'
        else mask := mask + '1';
        if abs(p-p7)>differenceIntensity then  mask := mask + '0'
        else mask := mask + '1';
        if abs(p-p8)>differenceIntensity then  mask := mask + '0'
        else mask := mask + '1';
        massMask[i,j] := mask;
        mask := '';
        p := 0;
        p1 := 0;
        p2 := 0;
        p3 := 0;
        p4 := 0;
        p5 := 0;
        p6 := 0;
        p7 := 0;
        p8 := 0;
        k := 0;
        k1 := 0;
        k2 := 0;
        k3 := 0;
        k4 := 0;
        k5 := 0;
        k6 := 0;
        k7 := 0;
        k8 := 0;
      end;
  end;
```

```
        SetLength(massBoolean, image.Width, image.Height);

         count := 0;
         for i:= pointSize to image.Width- 2 * pointSize do
          for j:= pointSize to image.Height- 2 * pointSize do
            begin
              str := massMask[i,j];
                for  z := 1 to 7 do
                  begin
                    if str[z] <> str[z+1] then Inc(count);
                  end;
               if count = 2 then
                begin
                  massBoolean[i,j] := 1;
                  count := 0;
                end
               else
                begin
                  massBoolean[i,j] := 0;
                  count := 0;
                end;
            end;

        for i:= pointSize to image.Width - 2 * pointSize do
          for j:= pointSize to image.Height - 2 * pointSize do
            begin
              if massBoolean[i,j] = 0 then
                begin
                  image.PutPixel(i,j,RGB(image.GetPixel(i, j).R,
image.GetPixel(i, j).G, image.GetPixel(i, j).B));
                end
              else
              begin
                image.PutPixel(i,j,RGB(255,255,255));
              end;
            end;

      var st1 := GetCurrentDir+'\out\'+filename;
      image.Save(st1);
    end;
  writeln('done');
    end.
```