harshad_v3.pas

```
PROGRAM numeros_Harshad (input, output);
var numInicial, numFinal, i, n, sumaDigitos : integer;
    encontrados: boolean;
begin
  writeln('especifique el intervalo de números naturales a probar: ');
  readln(numInicial, numFinal);      encontrados := false;
  for i:=numInicial to numFinal do begin
    n := i;   sumaDigitos := 0;
    while n > 0 do begin { sumar una cifra y quitarla }
      sumaDigitos := sumaDigitos + n mod 10;      n := n div 10
    end;
    if i mod sumaDigitos = 0 then begin  { i es un número 10-Harshad }
      if encontrados then write(', ') else encontrados := true;
      write(i:1)
    end
  end
end.
```

generarFichRectangulos_v4.pas

```
PROGRAM generarRectangulos (input, output);
type    tpVertice = record
                      x, y: real  { coordenadas X,Y del vértice }
                    end;
        tpCuadrilatero = record
                           v1, v2, v3, v4: tpVertice; { vértices del cuadrilátero }
                         end;
        tpFichCuadrilateros = file of tpCuadrilatero;
```

```
function esRectangulo(Q: tpCuadrilatero): boolean;
begin
  esRectangulo:=(Q.v1.y=Q.v2.y) and (Q.v3.y=Q.v4.y) and (Q.v1.x=Q.v4.x) and (Q.v2.x=Q.v3.x)
end;
```

```
procedure defRectangulo(var R: tpCuadrilatero; p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y: real);
begin
  R.v1.x:=p1x; R.v1.y:=p1y;   R.v2.x:=p2x; R.v2.y:=p2y;
  R.v3.x:=p3x; R.v3.y:=p3y;   R.v4.x:=p4x; R.v4.y:=p4y
end;
```

```
var Q, R: tpCuadrilatero;
    Cx, Cy: real; { coordenadas del centro del rectángulo }
    fCuad, fRect: tpFichCuadrilateros;
begin
  assign(fCuad, 'cuadrilateros.dat');  reset(fCuad);
  assign(fRect, 'rectangulos.dat');    rewrite(fRect);
  while not eof(fCuad) do begin
    read(fCuad, Q);
    if esRectangulo(Q) then begin
      Cx := (Q.v1.x+Q.v3.x)/2;     Cy := (Q.v1.y+Q.v3.y)/2;
      defRectangulo(R, Q.v1.x, Q.v1.y, Cx, Q.v2.y, Cx, Cy, Q.v4.x, Cy);  write(fRect, R);
      defRectangulo(R, Cx, Q.v1.y, Q.v2.x, Q.v2.y, Q.v2.x, Cy, Cx, Cy);  write(fRect, R);
      defRectangulo(R, Cx, Cy, Q.v2.x, Cy, Q.v3.x, Q.v3.y, Cx, Q.v3.y);  write(fRect, R);
      defRectangulo(R, Q.v1.x, Cy, Cx, Cy, Cx, Q.v3.y, Q.v4.x, Q.v4.y);  write(fRect, R)
    end
  end;
  close(fCuad);  close(fRect)
end.
```

filtro_caja_v1.pas

```
const MXDIM = 1000;

type  tpPixel  = 0..255;
      tpImagen = record
                   pix : array[1..MXDIM,1..MXDIM] of tpPixel;
                   ancho, alto: integer
                 end;
```

```
procedure filtroCaja(original: tpImagen; var filtrada: tpImagen);
{ Pre:  original contiene la imagen a filtrar
  Post: filtrada contiene el resultado de aplicar el filtro de caja a original
}
var     fil, col, f, c, fIni, fFin, cIni, cFin, suma : Integer;
begin
  filtrada.alto  := original.alto;
  filtrada.ancho := original.ancho;
  for col := 1 to filtrada.ancho do begin
    if col > 1              then cIni := col-1 else cIni := col;
    if col < filtrada.ancho then cFin := col+1 else cFin := col;
    for fil := 1 to filtrada.alto do begin
      if fil > 1             then fIni := fil-1 else fIni := fil;
      if fil < filtrada.alto then fFin := fil+1 else fFin := fil;
      suma := 0;
      for c := cIni to cFin do
        for f := fIni to fFin do suma := suma + original.pix[c, f];
      filtrada.pix[col, fil] := suma div ((cFin-cIni+1)*(fFin-fIni+1))
    end
  end
end;
```