Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1474

Examen de Teoría

10 de septiembre de 2020

Fundamentos de Informática
Grado en Tecnologías Industriales
Duración: 3 horas

Ejercicio 1 **[3.5 puntos]**

```
1  program regular;
2  uses math; {Para poder usar la funcion arccos}
3  type
4      tpVertice = record
5          x,y:Real;
6      end;
7      tpPolilinea = file of tpVertice;
8
9  function longitud(v1,v2:tpVertice):Real;
10 begin
11     longitud := sqrt((v2.x - v1.x)*(v2.x - v1.x) + (v2.y - v1.y)*(v2.y - v1.y));
12 end;
13
14 function angulo(v1,v2,v3:tpVertice):Real;
15 begin
16     angulo := arccos(((v2.x - v1.x)*(v3.x - v2.x) + (v2.y - v1.y)*(v3.y - v2.y))/
17                           (longitud(v1,v2)*longitud(v2,v3)));
18 end;
19
20
21 var
22     nombre : String;
23     v0, v1, vn, vn_1, vn_2 : tpVertice;
24     f : tpPolilinea;
25     longitudComun, anguloComun : Real;
26     poligono, equilatero, equiangulo : Boolean;
27 begin
28     write('Nombre del fichero : ');
29     readln(nombre);
30     assign(f,nombre);
31     reset(f);
32     read(f,vn_2, vn_1, vn);
33     v0 := vn_2; v1 := vn_1;
34     longitudComun := longitud(vn_2,vn_1);
35     equilatero := (longitud(vn,vn_1) = longitudComun);
36     anguloComun := angulo(vn_2,vn_1,vn);
37     equiangulo := true;
38
39     while not eof(f) do begin
40         vn_2 := vn_1;
41         vn_1 := vn;
42         read(f,vn);
43         if longitud(vn_1,vn) <> longitudComun then
44             equilatero := false;
45         if angulo(vn_2,vn_1,vn) <> anguloComun then
46             equiangulo := false;
47     end;
```

```
48
49      poligono := (longitud(vn,v0) = 0);
50      { Si no es cerrado no podemos comparar la ultima arista con la primera }
51      if poligono and (angulo(vn_1,v0,v1) <> anguloComun) then
52          equiangulo:=false;
53
54      if poligono and equilatero and equiangulo then
55          writeln('Poligono regular')
56      else begin
57          if not poligono then writeln('No es un poligono');
58          if not equilatero then writeln('No es equilatero');
59          if not equiangulo then writeln('No es equiangulo');
60      end;
61  end.
```

Ejercicio 2 **[3.5 puntos]**

```
program distanciamiento;

const
  NFIL = 10;
  NCOL = 15;

type
  tpButaca = record
    vacio: boolean;
    masc: boolean;
  end;

  tpPatioButacas = array[1..NFIL,1..NCOL] of tpButaca;

function valido(f: integer; c: integer): boolean;
begin
  valido := (f >= 1) and (f <= NFIL) and (c >= 1) and (c <= NCOL);
end;

function correcta(patio: tpPatioButacas): boolean;
var
  i, j: integer;
  k, l: integer;
  contPer, contPos, max: integer;

begin
  correcta := true;
  i := 1;
  while (i <= NFIL) and correcta do
  begin
    j := 1;
    while (j <= NCOL) and correcta do
    begin
      contPos := 0;
      contPer := 0;
      { Para cada butaca ocupada, miramos si se cumplen las condiciones }
      if not patio[i,j].vacio then
      begin
        for k:=i-2 to i+2 do
          for l:=j-2 to j+2 do
            if valido(k,l) then
            begin
              contPos := contPos + 1;
              if not patio[k,l].vacio then
              begin
                contPer := contPer + 1;
                if not patio[i,j].masc and not ((i=k) and (j=l)) then
                  correcta := false;
              end;
            end;
        max := contPos div 3;
        if contPer > max then
          correcta := false;
      end;
      j := j + 1;
    end;
    i := i + 1;
  end;
```

```
59 end;
60
61 procedure inicializarPatio(var patio: tpPatioButacas);
62 { Inicializa a un patio vacío (y sin mascarillas) }
63 var
64   i, j: integer;
65 begin
66   for i:=1 to NFIL do
67     for j:=1 to NCOL do
68     begin
69       patio[i,j].vacio := true;
70       patio[i,j].masc := false;
71     end;
72 end;
73
74 var
75   pat: tpPatioButacas;
76   f, c: integer;
77
78 begin
79
80   inicializarPatio(pat);
81   for f:= 1 to 5 do
82     for c:= 1 to 5 do
83     begin
84        {pat[2,c].vacio := false; }
85        pat[3,3].vacio := false;
86        {pat[2,c].masc := true; }
87     end;
88
89
90
91   writeln(correcta(pat));
92 end.
```

Ejercicio 3 **[3.0 puntos]**

```
program polinomios;
uses math; {Para poder usar la funcion power}

const
    MAXGRADO = 15;
    ERROR = 0.0001;
type
    tpPoly = array[0..MAXGRADO] of real;

function poly_eval(const p : tpPoly; x: real): real;
var
    res : real;
    i : integer;
begin
    res := p[0];
    for i:=1 to MAXGRADO do
        res := res + p[i]*power(x,i);
    poly_eval := res;
end;

function poly_zero(const p: tpPoly; a,b: real): real;
var
    an,bn : real;
    pa,pb : real;
    m,pm : real;
begin
    an := a;
    pa := poly_eval(p,an);
    bn := b;
    pb := poly_eval(p,bn);
    while abs(an-bn)>ERROR do
    begin
        m := (an+bn)/2;
        pm := poly_eval(p,m);
        if (pa*pm)>=0
        then begin
            an := m;
            pa := pm;
        end
        else begin
            bn := m;
            pb := pb;
        end;
    end;
    poly_zero := (an+bn)/2;
end;

begin
end.
```