```
Program Tesina2;

uses crt;

const dim = 10; nnodi = 20;

type filo = record x1,y1,x2,y2,maxy,miny:integer; way:char; end;
     link = ^nodo;
     nodo = record inf,val:integer; pr:link; end;
     vet = array [-dim..dim] of integer;
     circuito = array [1..dim] of filo;
     griglia = array [1..dim,1..dim] of integer;
     lista = array [-dim..dim] of link;

var w,padre,dfn,udfn,low,color  : vet;
    u,v                         : circuito;
    g                           : griglia;
    i,j,k,l,f,n,x,y             : integer;
    in1,in2,fn1,fn2             : integer;
    s,top,comp                  : lista;
    p,q                         : link;
    compatib                    : boolean;
    ctr                         : array [-dim..dim] of boolean;


label fine;


   Function min(x,y:integer):integer;
     begin
       if x<y then min:=x
              else min:=y;
     end;

   Function max(x,y:integer):integer;
     begin
       if x>y then max:=x
              else max:=y;
     end;
```

```
Procedure Posizione (w:circuito; a,b:integer; var c:griglia);  { a.x1 < b.x1 }
  begin

    if (w[a].x2 < w[b].x1)
      or (w[a].maxy < w[b].miny)
      or (w[b].maxy < w[a].miny)                        { fili indipendenti }
        then begin c[a,b]:=0; c[b,a]:=0; end else

    if (w[a].x2 > w[b].x2)
      and (w[a].maxy > w[b].maxy)
      and (w[a].miny < w[b].miny)                       { b interno ad a }
        then begin c[a,b]:=0; c[b,a]:=0; end else

    if (w[a].x2 > w[b].x2)
      and (w[a].maxy < w[b].maxy)
      and (w[a].miny > w[b].miny)
        then begin c[a,b]:=-1; c[b,a]:=-1; end else     { circuito non valido }

    if (w[a].x2 < w[b].x2)
      and ( (w[a].y1 - w[b].y1) * (w[a].y2 - w[b].y2) > 0 )   then
          begin
          if ( (w[a].y1 - w[b].y1) * (w[a].y1 - w[b].y2) < 0 )
            then begin c[a,b]:=1; c[b,a]:=1; end      { a or not b }  {il caso "b or not a" non può mai avvenire}
            else begin c[a,b]:=2; c[b,a]:=2; end;     {(a or b) and (not a or not b)}
          end               else

    if (w[a].x2 < w[b].x2)  then
          begin
          if ( (w[a].y2 > w[b].maxy) or (w[a].y2 < w[b].miny) )
            then begin c[a,b]:=3; c[b,a]:=3; end      { a }
            else begin c[a,b]:=4; c[b,a]:=4; end      { not b }
          end               else

    if (w[a].x2 > w[b].x2)  then
          begin
          if ( (w[a].y1 > w[b].maxy) or (w[a].y1 < w[b].miny) )
            then begin c[a,b]:=5; c[b,a]:=5; end      { not a }

            else begin c[a,b]:=3; c[b,a]:=3; end      { a }
          end;
  end;
```

```
Procedure creagrafo(var v:lista; f:integer);
  var   i,j:integer;                          {estremi degli archi}
        p:link;
  begin
    for k:=-f to f do v[k]:=nil;                 {inizializzazione di v}
    for i:=1 to f do
    for j:=i+1 to f do
      begin
        case g[i,j] of          { calcola il num di spigoli }
          0 : ;
          1 : begin
                new(p); p^.inf:=i; p^.val:=-1; p^.pr:=v[j]; v[j]:=p;
                new(p); p^.inf:=-j; p^.val:=-1; p^.pr:=v[-i]; v[-i]:=p;
              end;
          2 : begin
                new(p); p^.inf:=i; p^.val:=-1; p^.pr:=v[-j]; v[-j]:=p;
                new(p); p^.inf:=j; p^.val:=-1; p^.pr:=v[-i]; v[-i]:=p;
                new(p); p^.inf:=-i; p^.val:=-1; p^.pr:=v[j]; v[j]:=p;
                new(p); p^.inf:=-j; p^.val:=-1; p^.pr:=v[i]; v[i]:=p;
              end;
          3 : begin
                new(p); p^.inf:=i; p^.val:=-1; p^.pr:=v[-i]; v[-i]:=p;
              end;
          4 : begin
                new(p); p^.inf:=-j; p^.val:=-1; p^.pr:=v[j]; v[j]:=p;
              end;
          5 : begin
                new(p); p^.inf:=-i; p^.val:=-1; p^.pr:=v[i]; v[i]:=p;
              end;
        end;
      end;
  end;


Procedure attcoda(z,y:integer);  {attacca la lista di z in quella di y}
  var p:link;
    begin
      top[y]^.pr:=comp[z];  top[y]:=top[z];
      ctr[z]:=false;        ctr[y]:=true;
    end;
```

```
Procedure DFS(z:integer);
  var p:link; w:integer;
  begin
    color[z]:=1;    x:=x+1;
    dfn[z]:=x; udfn[x]:=z; low[z]:=x;
    p:=s[z];
    while (p<>nil) do
      begin
        w:=p^.inf;
        if (color[w]=0)
          then begin padre[w]:=z; DFS(w); low[z]:= min(low[z],low[w]); end
          else
        if (color[w]=1)
          then begin low[z]:=min(low[z],low[w]); low[padre[z]]:=min(low[z],low[padre[z]]); end
          else
        if (color[udfn[low[w]]]=1)
          then low[z]:=min(low[z],low[w]);
        {if (color[w]=2) and (color[udfn[low[w]]]=2) then - si ha un arco tra due c.f.c. distinte}
        p:=p^.pr;
      end;
    if low[z]<dfn[z] then attcoda(z,udfn[low[z]])    { z = puo raggiungere nodi + alti }
                     else begin                      { z = nodo + alto = rappr. di una c.f.c. }
                            p:=comp[z];

                            repeat
                              w:=p^.inf;
                              if (s[w]^.val= -1) then
                                begin s[w]^.val:=0; s[-w]^.val:=1; end
                                else
                              if (s[w]^.val <> s[comp[z]^.inf]^.val) then
                                begin
                                  writeln('Errore: il filo ',abs(w),' non può essere accettato.');
                                  compatib:=false;
                                end;
                              p:=p^.pr
                            until (p=nil) or (not compatib);
                          end;
    color[z]:=2;
  end;

{-------------------------------------------------------------------------------------------------------------------------------}
```

```
Begin  {main}

{Passo 1 : inserimento dei fili}

  write('Quanti fili devono essere sistemati ? ');  readln(f);

  writeln('Inserire i due estremi per ogni filo (le coordinate devono essere comprese tra 1 e ',dim,').');
  for i:=1 to f do
    begin
      Repeat
        writeln(i,'^ filo.');
        write('   1^ vertice - ascissa  : ');      readln(v[i].x1);
        write('              - ordinata : ');      readln(v[i].y1);
        write('   2^ vertice - ascissa  : ');      readln(v[i].x2);
        if (v[i].x2 < v[i].x1) then writeln('filo non valido!');
      until (v[i].x2 >= v[i].x1);
      write  ('              - ordinata : ');      readln(v[i].y2);
      v[i].maxy:=max(v[i].y1,v[i].y2);
      v[i].miny:=min(v[i].y1,v[i].y2);
    end;

{Passo 2 : ordinamento dei fili}

  for i:=1 to dim do w[i]:=0;
  for i:=1 to f do w[v[i].x1]:=w[v[i].x1]+1;
  for i:=2 to dim do w[i]:=w[i]+w[i-1];
  for i:=f downto 1 do begin u[w[v[i].x1]]:=v[i]; w[v[i].x1]:=w[v[i].x1]-1; end;
      {u è un vettore ordinato per x1 crescente}

  for i:=1 to f do  for j:=1 to f do g[i,j]:=9;   {alla fine solo se i=j si avrà g[i,j]=9}

{Passo 3 : classificazione delle relazioni - si creano le clausole}

  for i:=1 to f do
  for j:=i+1 to f do
    begin
      posizione(u,i,j,g);                                          { classifica le relazioni nella matrice g }
      if (g[i,j] = -1)
         then begin writeln('Errore: la coppia di fili (',i,',',j,') non può essere accettata.'); goto fine; end;
    end;
```

```
{Passo 4 : viene creato il grafo delle clausole}

  creagrafo(s,f);                                                    { s = lista di adiacenza }

{Passo 5 : studio delle c.f.c. del grafo ed effettua un T-sort sui rappresentanti}

  x:=0; compatib:=true;
  for k:=-f to f do                                                  { inizializza i vettori d'appoggio }
    begin
      color[k]:=0; padre[k]:=0;
      ctr[k]:=true;
      new(q); comp[k]:=q; top[k]:=q;
      top[k]^.inf:=k; top[k]^.pr:=nil;
    end;

  writeln;
  for k:=-f to f do if (color[k]=0) and (k<>0)   then  DFS(k);       { visita in prof. }
  if (not compatib) then goto fine;

  for k:=-f to f do                                                  {dà in uscita le c.f.c.}
    if (ctr[k]) and (k<>0)  then
        begin
          q:=comp[k];
          repeat
            x:=q^.inf;
            if (x>0) then
              if s[x]^.val=1 then u[x].way:='v'
                             else u[x].way:='o';
            q:=q^.pr;
          until  (q=nil);
        end;

{Passo 5 : assegnamo la way del filo in base al T / F dei nodi}

  writeln;  writeln('Percorsi compatibili :');
  for i:=1 to f do writeln(i,'^ filo (',u[i].x1,'-',u[i].y1,'): percorso ',u[i].way);

  fine:   writeln;
          write('  - Programma terminato. -');    readln;
End.
```