```
PROGRAM Chiusura_transitiva;

uses crt;

const dim=30;

type matr  = array [1..dim,1..dim] of boolean;
     vet   = array [1..dim] of integer;
     vet2  = array [1..dim,1..2] of integer;
     vetb  = array [1..dim] of boolean;
     link  = ^nodo;
     nodo  = record inf:integer; pr:link; end;
     lista = array [1..dim] of link;

var  i,j,k,x,y,m,n : integer;
     g             : matr;
     col,v         : vet;
     w             : vet2;
     b             : vetb;
     s             : lista;

  Procedure creamatr(var g:matr; n,m:integer);                  { crea una matrice di adiacenza n x n }
    var i,j,k:integer;
    begin
      for i:=1 to n do
        for j:=1 to n do g[i,j]:=false;
      for k:=1 to m do
        begin
          writeln('  > ',k,'^ vincolo.');
          write  ('    - 1^ nodo : ');     readln(i);
          write  ('    - 2^ nodo : ');     readln(j);
          g[i,j]:=true;
        end;
    end;

  Procedure Crealista(g:matr; n:integer; var s:lista);          { crea un vettore di n liste di adiacenza }
    var i,j:integer;
        p:link;
    begin
      for i:=1 to n do s[i]:=nil;
      for i:=1 to n do
      for j:=1 to n do if g[i,j] then
        begin new(p); p^.inf:=j; p^.pr:=s[i]; s[i]:=p; end;
    end;
```

```
Procedure chiusura(x:integer);                          { se esiste un arco (x,y) ed esiste un arco (y,z) }
  var y,z:integer; p:link;                              { allora viene creato l'arco (x,z) }
  begin                                                 { inoltre, ∀ nodo x viene creato l'arco (x,x) }
    g[x,x]:=true;      p:=s[x];
    while (p<>nil) do
      begin
        y:=p^.inf;
        for z:=1 to n do if g[y,z] then g[x,z]:=true;
        p:=p^.pr;
      end;
  end;

Procedure VIP(x:integer);                               { visita in profondità sulle liste di adiacenza }
  var p:link; y:integer;                                { viene creato un vettore di ordinamento topologico }
  begin
    col[x]:=1;      p:=s[x];
    while (p<>nil) do
      begin
        y:=p^.inf;
        if (col[y]=0) then VIP(y);
        p:=p^.pr;
      end;
    v[j]:=x;      j:=j-1;
  end;

Procedure TMatching(var w:vet2; var k:integer);         { crea un matching studiando la matrice di adiacenza }
  var i,j:integer;       b:vetb;                        { seguendo l'ordinamento topologico }
  begin
    for i:=1 to n do b[i]:=false;
    i:=1; k:=1;
    while (i<n) do
      begin
        if (not b[v[i]]) then
          begin
            j:=i+1;
            while (b[v[j]]) and (j<=n) do j:=j+1;
            if (j<=n) and (not g[v[i],v[j]]) and (not g[v[j],v[i]]) then
              begin w[k,1]:=v[i]; w[k,2]:=v[j]; k:=k+1; b[v[i]]:=true; b[v[j]]:=true; end;
          end;
        i:=i+1;
      end;
  end;
```

```
{ Inizio programma generale }

Begin
  clrscr;
  writeln('Programma : Tesina n^3');
  write  (' * numero di lavori  : '); readln(n);
  write  (' * numero di vincoli : '); readln(m);

  creamatr(g,n,m);                                    { memorizza il grafo su una matrice di adiacenza }

  crealista(g,n,s);                                   { crea le liste di adiacenza }

  for i:=1 to n do col[i]:=0;
  j:=n;
  for i:=1 to n do if (col[i]=0) then VIP(i);         { visita in prof per oridinare i vertici }

  for i:=n downto 1 do chiusura(v[i]);                { chiusura transitiva del grafo }

  tmatching(w,k);                                     { matching mxl che rispetta l'ordinamento }

  writeln('possibili accoppiamenti tra i lavori:');
  for i:=1 to k-1 do writeln(w[i,1],' - ',w[i,2]);    { restituisce in output il matching ottenuto }
  readln;
End.
```