```
program Detectie_lista_circulara;

uses Crt;

type
    { Definim tipurile de date pentru un nod al listei. Un nod
      contine o informatie de tip Integer si un pointer care pastreaza
      adresa nodului urmator din lista. Daca nu mai exista un nod urmator
      in lista, atunci pointerul are valoarea "nil". }
    PNod = ^TNod;
    TNod = record
          info : Integer;
          urm : PNod;
    end;

var
   { Memoram doar primul nod din lista. Pornind de la acest nod putem
     parcurge oricand intreaga lista. }
   start : PNod;


{ Procedura care insereaza un nod la inceputul listei. Procedura
  primeste ca si parametru valoarea intreaga ce se memoreaza in noul nod. }
procedure InserareInceput(info : Integer);
var nod_nou : PNod;
begin
     { Inainte de orice alocam memorie in HEAP pentru noul nod. }
     New(nod_nou);

     { Apoi memoram in noul nod valoarea intreaga ce a fost trimisa ca si
       parametru. }
     nod_nou^.info := info;

     { Legam noul nod de inceputul listei. }
     nod_nou^.urm := start;

     { Si modificam inceputul listei. Noul nod va fi de acum inceputul
       listei. }
     start := nod_nou;
end;


{ Procedura care afiseaza continutul listei. Atentie: daca lista este
  circulara atunci aceasta procedura intra in ciclu infinit. }
procedure Afisare;
var cursor : PNod;
begin
     { Folosim un cursor care viziteaza pe rand fiecare nod din lista.
       Se porneste cu cursorul de la nodul de start. }
     cursor := start;

     { Cat timp nu am ajuns la "nil", adica la sfarsitul listei. }
     while cursor <> nil do
           begin
                { Afisam valoarea din nodul indicat de cursor. }
                Write(cursor^.info, ' ');

                { Dupa care mutam cursorul la urmatorul nod. }
                cursor := cursor^.urm;
           end;

     WriteLn;
end;
```

```
{ Procedura care transforma o lista liniara intr-o lista circulara.
  Se conecteaza ultimul nod din lista la primul nod din lista. }
procedure ConecteazaCircular;
var ultimul : PNod;
begin
     { Intai gasim care este ultimul nod din lista. Ultimul nod din lista
       este acel nod care nu mai are succesori. Pornim cautarea de la
       primul nod din lista si atat timp cat nodul curent are succesor
       avansam in lista. }
     ultimul := start;
     while ultimul^.urm <> nil do
           ultimul := ultimul^.urm;

     { Odata ajunsi la ultimul nod, il conectam de primul nod din lista.
       Din acest moment lista a devenit circulara. }
     ultimul^.urm := start;
end;


{ Functie care detecteaza daca o lista este liniara sau circulara.
  Se foloseste algoritmul lui Tarjan:
  * se folosesc doua noduri cursor care ambele pornesc de la primul nod
    din lista;
  * unul din cele doua noduri cursor inainteaza in lista cu cate un nod
    la un moment dat, iar celalalt nod cursor inainteaza in lista cu cate
    doua noduri la un moment dat;
  * daca lista este liniara, atunci cursorul care inainteaza tot cu cate
    doua noduri va ajunge la valoarea "nil";
  * daca lista este circulara, atunci cursorul care inainteaza tot cu cate
    doua noduri va ajunge din urma cursorul care inainteaza cu un singur
    nod. }
function EsteCirculara:Boolean;
var unu, doi : PNod;
begin
     { Cele doua noduri cursor pornesc de la primul nod din lista. }
     unu := start;
     doi := start;

     repeat
           { Daca nu s-a ajuns la sfarsitul listei, primul cursor
             avanseaza cu un nod in lista. }
           if unu <> nil then
              unu := unu^.urm;

           { Daca nu s-a ajuns la sfarsitul listei, al doilea nod
             cursor inainteaza cu doua noduri in lista. }
           if doi <> nil then
              doi := doi^.urm;
           if doi <> nil then
              doi := doi^.urm;

           { Repetam acestea operatii pana cand fie al doilea cursor
             ajunge la "nil", fie al doilea cursor il ajunge din urma
             pe primul. }
     until (doi = nil) or (doi = unu);

     { Daca al doilea cursor a ajuns la "nil" atunci lista este liniara. }
     if doi = nil then
        EsteCirculara := false
     { Altfel lista este circulara. }
     else
```

```
          EsteCirculara := true;
end;


begin
     ClrScr;

     { Cream o lista cu cateva noduri oarecare. }
     start := nil;
     InserareInceput(10);
     InserareInceput(14);
     InserareInceput(5);
     InserareInceput(20);
     InserareInceput(45);
     InserareInceput(34);

     { Afisam lista si verificam daca este circulara. Vom vedea
       ca lista nu este circulara, ci este liniara. }
     Afisare;
     WriteLn( EsteCirculara );

     { Transformam lista in una circulara. }
     ConecteazaCircular;

     { Atentie: Daca acum apelam procedura de afisare se va intra
       in ciclu infinit. Vom apela doar functia de verificare daca
       lista este circulara. Vom vedea ca lista este intradevar
       circulara. }
     {Afisare;}
     WriteLn( EsteCirculara );
end.
```