Îndrumar pentru laboratorul de informatică aplicată. Probleme rezolvate in Turbo Pascal

Conf.dr.inf. Raluca Giurma-Handley

Prof.dr.ing. Dorin Cotiușcă-Zaucă

Șef lucrări dr.ing. Marius Telișcă

Indrumar pentru laboratorul de informatica aplicată. Probleme rezolvate in Turbo Pascal

1. - Mediul de programare Turbo Pascal

Mediul de programare Turbo Pascal este dedicat dezvoltării programelor scrise în limbajul Pascal. Dintre componentele sale menţionăm:

- editorul de programe -acesta este un editor de texte specializat, conceput astfel încât să uşureze dactilografierea şi modificarea programelor sursă Pascal;

- componenta de gestiune a fişierelor, care asigură, în principal, salvarea în fişiere pe disc a programelor editate şi încărcarea programelor sursă de pe disc;

- compilatorul, care analizează corectitudinea sintactică a programului editat şi generează forma sa executabilă;

- componenta ce controlează execuţia programului, cu revenire la editorul de programe în momentul terminării normale a programului sau la producerea unei erori;

- sistemul de informaţii ajutătoare (HELP), care asigură obţinerea de informaţii referitoare la utilizarea mediului de programare Turbo Pascal şi, respectiv, la elementele limbajului Pascal.

Începerea sesiunii de lucru Turbo Pascal

O sesiune de lucru Turbo Pascal este declanşată de comanda *turbo*. Aceasta se poate introduce într-o fereastră MS-DOS sau comanda se poate activa prin intermediul pictogramei Turbo Pascal corespunzatoare din Windows.

Ca efect al comenzii, pe ecran se afişează fereastra principală, peste care se suprapune o fereastră de editare (vezi figura 1).

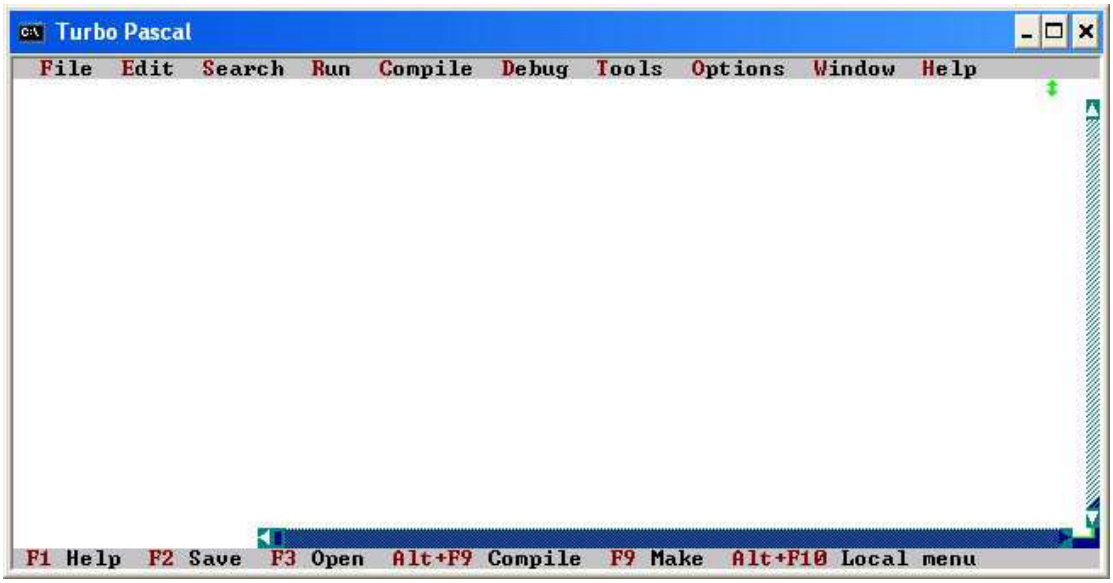


Figura 1. Fereastră de editare

În partea superioară a ferestrei principale se află meniul principal, care permite accesul la sistemul de meniuri al mediului de programare Turbo Pascal. Deoarece comenzile şi opţiunile Turbo Pascal sunt foarte numeroase, ele sunt grupate în funcţie de semnificaţie. Comenzile dintr-un grup sunt prezentate în cadrul unui meniu specific şi pot fi selectate atunci când meniul corespunzător este afişat pe ecran şi comanda respectivă are sens (de exemplu nu se poate solicita execuţia operaţiei "salvare fişier" atunci când nu există un fişier în curs de prelucrare).

Anumite comenzi pot fi însă transmise direct, prin acţionarea unei taste sau a unei combinaţii de taste de activare (hot keys). Din această categorie fac parte şi cele câteva menţionate în ultima linie din fereastra principală (numită linie de Informare), care au următoarele semnificaţii:

F1 Help	-acces la sistemul de informaţii ajutătoare;
F2Save	-salvarea programului din fereastra de editare într-un fişier pe disc;
F3 Open	-deschiderea unei ferestre de editare în care se încarcă conţinutul unui fişier sursă;
Alt F9 Compile	-compilarea programului din fereastra de editare;
F9 Make	-generarea formei executabile a programului;
F10 Local Menu	-acces la meniul principal, pentru a opera cu sistemul de meniuri Turbo Pascal.

Principalele elemente ale ferestrei de editare sunt:

-numele fişierului sursă în care se salvează textul programului editat; dacă nu a fost ales încă un nume pentru fişierul sursă, atunci numele afişat este NONAME00.PAS;

-numărul de ordine al ferestrei de editare curente (mediul Turbo Pascal poate opera cu mai multe ferestre de editare dar, pentru simplitate, în cele ce urmează vom trata numai cazul utilizării unei singure ferestre de editare);

-poziţia curentă în cadrul textului sursă, specificată în partea de jos a ferestrei, sub forma Indice_linie: Indice_coloană;

-zona rezervată afişării conţinutului fişierului sursă, situată în interiorul cadrului ferestrei.

Salvarea pe disc a programului sursă

Deşi nu este obligatoriu, recomandăm ca în cazul editării unui nou program să se fixeze numele fişierului sursă corespunzător chiar de la începutul editării. Aceasta se realizează prin intermediul comenzii de salvare (Save), transmisă direct mediului Turbo Pascal prin apăsarea tastei F2. în acest caz se afişează o fereastră care permite programatorului să specifice numele fişierului sursă. Dacă se introduce numele test, mediul de programare îi adaugă automat extensia .pas, deci identificatorul complet al fişierului sursă va fi test.pas.

După introducerea numelui fişierului, la apăsarea tastei ENTER, se revine la fereastra de editare şi se poate trece la editarea textului programului.

Se recomandă, ca pe parcursul editării textului programului, acesta să fie salvat din când în când, pentru a evita situaţiile neplăcute, de genul „cădere de tensiune" sau „blocare sistem", în urma cărora textul editat se pierde şi este necesară reintroducerea sa.

Editarea unui program sursă Pascal

Introducerea textului unui program se face caracter cu caracter. Poziţia următorului caracter introdus este pusă în evidenţă printr-un marcaj clipitor, numit cursor.

Pentru a termina o linie şi a trece la linia următoare se apasă tasta ENTER. Editorul Turbo Pascal indentează automat liniile introduse, plasând cursorul sub primul caracter diferit de blanc din linia terminată (spre deosebire de alte editoare, care îl plasează la marginea din stânga a ecranului).

Pentru poziţionarea la capătul din stânga al liniei trebuie apăsată tasta HOME. Aceasta

face parte din grupul tastelor care comandă deplasarea rapidă în cadrul textului editat, după cum urmează:

CTRL+A-salt la începutul cuvântului anterior;

CTRL+F-salt la începutul cuvântului următor;

CTRL+Q.E-salt la prima linie de pe ecran;

CTRL+Q.X-salt la ultima linie de pe ecran;

HOME-salt la începutul liniei curente;

END-salt la sfârşitul liniei curente;

PAGE UP-salt la pagina anterioară;

PAGE DOWN-salt la pagina următoare.

Deplasările pas cu pas se realizează apăsând tastele direcţionale, marcate cu săgeţi.

Pe parcursul editării programului sursă este posibil să se greşească şi să fie necesară ştergerea unuia sau a mal multor caractere. în acest scop pot fi utilizate următoarele comenzi:

DELETE-şterge caracterul pe care este poziţionat cursorul;

BACKSPACE -această tastă se utilizează în mod obişnuit pentru ştergerea caracterului din stânga cursorului; efectul este diferit dacă la stânga cursorului există numai blancuri, caz în care ştergerea se efectuează până la nivelul de indentare anterior;

CTRL+T-şterge un cuvânt sau finalul acestuia, începând de la caracterul pe care este poziţionat cursorul;

CTRL+Q.Y-şterge toate caracterele de la dreapta cursorului;

CTRL+Y-şterge linia pe care este poziţionat cursorul.

De exemplu, pentru a înlocui cuvântul salut cu SALUT, se poate proceda astfel:

- se poziţionează cursorul la începutul cuvântului;
- se şterge cuvântul utilizând CTRL+T;
- se introduce noua formă a cuvântului.

Modul obişnuit de lucru al editorului de programe este modul „inserare". Aceasta înseamnă că în cazurile în care cursorul se află în interiorul unei linii, apăsarea unei taste determină inserarea caracterului corespunzător (caracterele de la dreapta cursorului deplasându-se corespunzător).

Modul de lucru al editorului poate fi comutat de la „Inserare" la „substituţie", apăsând tasta INSERT. în modul „substituţie", la apăsarea unei taste, caracterul indicat de cursor este în

locuit de cel introdus. Schimbarea modului de lucru al editorului este semnalată prin schimbarea formei cursorului.

Comanda prin care se solicită compilarea programului sursă şi generarea celui executabil este F9 (Make). în cazul programelor sursă fără erori suntactice se afişează o fereastră care furnizează informaţii despre rezultatul compilării.

Daca programul are erori, atunci compilatorul se opreşte la prima eroare întâlnită, redând controlul editorului, în acest caz cursorul este poziţionat în locul în care a fost detectată eroarea şi se afişează un mesaj referitor la cauza probabilă a erorii, ca în cazul programelor de dimensiuni mai mari se pot face compilări chiar înainte de editarea completă a textului programului sursă, pentru a verifica dacă porţiunea introdusă a fost corect dactilografiată.

Corectarea erorilor sintactice

Chiar dacă revenirea în etapa de editare se face cu poziţionarea cursorului în locul detectării erorii, cauza acesteia trebuie căutată cu atenţie, deoarece elementul greşit se poate afla într-o linie anterioară celei pe care este poziţionat cursorul.

După găsirea cauzei erorii şi modificarea programului este necesară o nouă compilare, procesul repetându-se 'până în momentul obţinerii unei program fără erori sintactice.

Execuţia programului

Cea mai simplă comandă de lansare în execuţie a programului editat este CTRL+F9. Ca rezultat, este afişată fereastra utilizator (în care se afişează mesajele scrise prin program şi ecoul datelor introduse de utilizator), iar apoi se dă controlul programului respectiv.

La încheierea execuţiei programului controlul este redat mediului Turbo Pascal, care reafişează fereastra de editare.

Astfel, în cazul execuţiei programului test, deoarece aceasta este practic instantanee, se remarcă numai o scurtă clipire a ecranului. Pentru a vedea ce anume s-a întâmplat, este necesară afişarea ferestrei utilizator, în acest scop se utilizează comanda ALT+F5. Pentru a reveni la fereastra TP se poate apăsa orice tastă.

În cazul exemplului considerat, o singură execuţie a programului este suficientă pentru a constata dacă acesta funcţionează corect, be cele maj, multe ori sunt însă necesare mai multe execuţii succesive, pentru diferite seturi de date. în procesul de testare a programului este posibil să fie detectate erori, în cazul unei erori detectate de mediul de programare, execuţia se termină

cu afişarea unui mesaj de eroare corespunzător. Alte erori sunt cele observate de utilizator, atunci când compară rezultatele afişate de program cu cele pe care le aştepta.

Corectarea erorilor impune modificări ale textului programului sursă, urmate de compilare şi execuţie.

Trecerea la editarea unui alt program

După obţinerea unui program care funcţionează conform aşteptărilor, se poate trece la editarea unui alt program. în acest scop se închide fereastra de editare curentă, prin comanda ALT+F3. Dacă varianta curentă a programului editat nu a fost deja salvată, atunci se afişează o fereastra care conţine trei butoane de comandă, dintre care poate fi selectat doar unul.

Un buton de comandă poate fi selectat în două moduri:

- prin poziţionarea pe buton folosind tasta TAB, urmată de apăsarea tastei ENTER;

- apăsând tasta corespunzătoare, dacă în textul înscris pe butonul respectiv este pusă în evidenţă o anumită literă. Efectul selectării este:

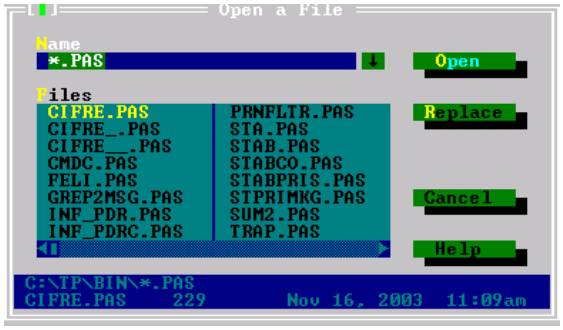
Yes-se memorează (salvează) versiunea curentă (cea mai recentă) a programului editat;

No-se păstrează versiunea salvată anterior, iar versiunea curentă se pierde;

Cancel-se revine la etapa de editare a programului.

Dacă este selectat butonul Yes sau butonul No, atunci fereastra de editare este închisă şi pe ecran rămâne numai fereastra principală.

Pentru a trece efectiv la editarea unui alt program, se utilizează comanda F3 (Open). Aceasta are ca efect afişarea ferestrei din figura



In acest moment se poate introduce un nume de fişier nou (care nu apare în lista afişată) sau se poate alege unul dintre fişierele existente, pentru a modifica sau re executa programul sursă pe care îl conţine.

Alegerea unui fişier existent se poate face în două moduri:

- prin introducerea numelui fişierului;

- apăsând mai întâi tasta TAB, după care banda luminoasă de selecţie trebuie poziţionată, folosind tastele direcţionale, pe numele fişierului dorit. .

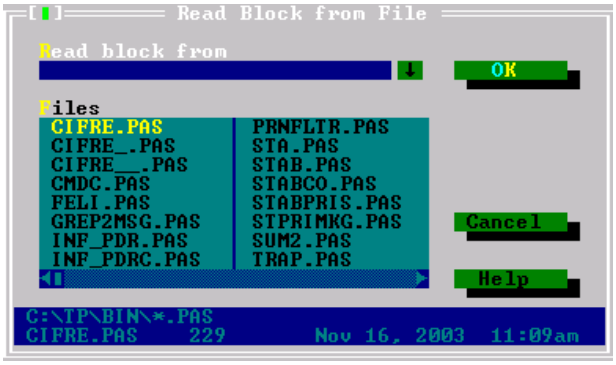
Selecţia devine efectivă în momentul apăsării tastei ENTER, moment în care se deschide d fereastră de editare în care apare conţinutul fişierului selectat.

Crearea unei variante a unui program existent

Dacă se doreşte crearea unei variante a unui program, cu păstrarea versiunii existente, se poate proceda în mai multe feluri, în cele ce urmează prezentăm o singură soluţie, destul de simplă:

- se trece la editarea unui nou fişier, în care se va păstra noua variantă a programului;

- se copiază varianta curentă a programului, folosind comanda CTRL+K,R; ca efect al acestei comenzi se afişează fereastra din figura



care permite selectarea fişierului din care se copiază, la fel ca în cazul încărcării unui fişier sursă existent;

- textul copiat din fişierul selectat este pus în evidenţă prin modul de afişare; pentru a trece la afişarea normală se utilizează comanda CTRL+K.H;

- se modifică programul sursă copiat şi se salvează în noul fişier.

Obţinerea informaţiilor ajutătoare

Informaţiile ajutătoare sunt grupate în ecrane de informare cu conţinut fix. Conţinutul ecranului de informare corespunzător contextului curent se afişează într-o fereastră-help. în funcţie de dimensiunile ecranului şi ale ferestrei, afişarea este integrală sau parţială. Pentru a aduce în cadrul ferestrei acele porţiuni din ecranul de informare care nu sunt vizibile, utilizatorul poate folosi tastele direcţionale (cele marcate cu săgeţi, precum și PAGE UP şi PAGE DOWN).

În majoritatea ecranelor de informare apar casete care conţin titlurile altor ecrane. Aceste casete pot fi selectate în mai multe moduri: utilizând tastele direcţionale, respectiv TAB şi SHIFT+TAB, sau introducând litera (literele) cu care începe textul din casetă. Odată selectată caseta dorită, apăsarea tastei ENTER determină afişarea ferestrei de informare corespunzătoare.

O sesiune de informare este lansată printr-una din următoarele comenzi:

ALT+H sau F10.H-determină afişarea meniului Help, din care pot fi selectate diferite ecrane de informare;

F1-are ca efect afişarea ecranului corespunzător contextului curent;

CTRL+F1-se utilizează în cazul în care pe parcursul editării este necesară obţinerea de informaţii ajutătoare cu privire la un element al limbajului, în astfel de cazuri este suficient să se deplaseze cursorul până la elementul respectiv (de exemplu un cuvânt cheie sau un operator) şi să se dea comanda CTRL+F1;

SHIFT+F1-determină afişarea ecranului de informare Index;

ALT+F1-în urma acestei comenzi se revine la fereastra de informare activată anterior, sau se afişează o primă fereastră.

Sesiunea de informare se încheie la apăsarea tastei ESC.

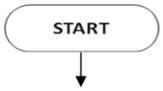
Incheierea sesiunii de lucru

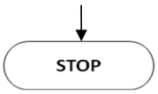
Pentru a încheia sesiunea de lucru Turbo Pascal se utilizează comanda ALT+X. în cazul în care programul sursă nu a fost salvat sau a suferit modificări după ultima salvare, în caz contrar TP propune salvarea acestuia.

2 - Descrierea algoritmilor prin scheme logice. Structura unui program in Turbo Pascal

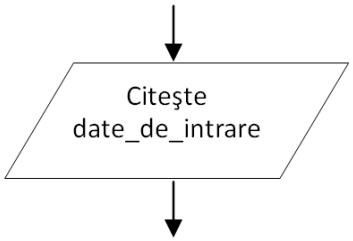
Schema logică este descrierea grafică a unui algoritm cu ajutotul unor simboluri care indică paşii algoritmului şi ordinea lor de execuţie. Fiecărei operaţii sau pas i se ataşează un simbol geometric în interiorul căruia se înscrie operaţia de executat. Succesiunea operaţiilor este simbolizată prin săgeţi. Schema logică reflectă de fapt modul de execuţie al programului corespunzător algoritmului.
Se utilizează următoarele simboluri grafice:

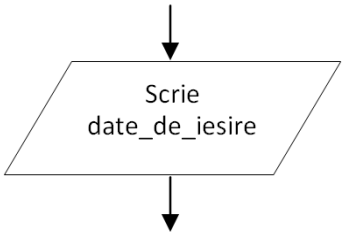
a) terminal-marchează începutul, sfârşitul sau un punct de revenire în schema logică, într-o schemă logică există un singur terminal de START și un singur terminal de STOP.



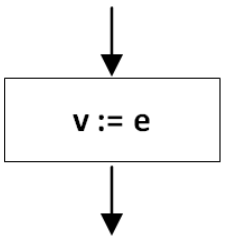


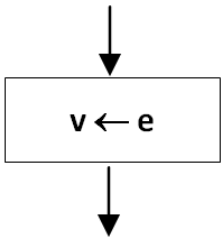
b) bloc de intrare/ieşire (citire/scriere)-permite precizarea datelor de intrare în problemă prin iniţializarea cu valori corespunzătoare a variabilelor care codifică aceste date sau permite afişarea unor rezultate (obţinute ca date de ieşire din algoritm).





c) atribuire sau bloc de calcul-reprezintă o operaţie de calcul cu atribuirea rezultatului variabilei din membrul stâng. Simbolul de atribuire poate fi: ← sau :=

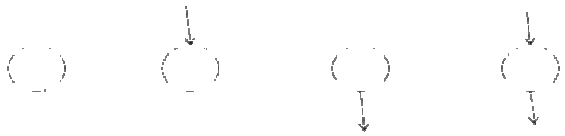




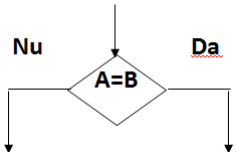
Variabilele din membrul drept al blocului de calcul trebuie să aibă valori bine definite (rezultate în urma citirii sau calculului); vechea valoare a variabilei din membrul stâng se pierde, peste ea scriindu-se noua valoare rezultată în urma operaţiei de calcul.

d) linia de flux-indică direcţia fluxului operaţiilor

e) conectorul-indică punctul în care o linie de flux se conectează la o alta linie sau o întrerupere a liniei de flux.



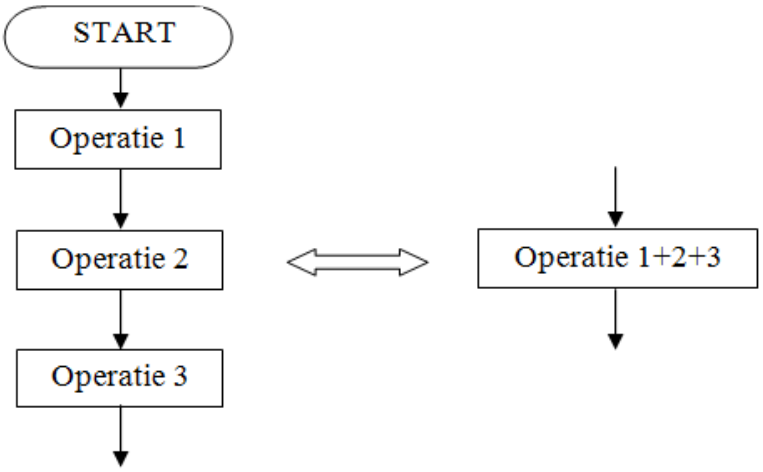
f) decizia logică-indică uri test logic al cărui rezultat poate fi adevărat (true) sau fals (false), continuarea fluxului de prelucrare făcându-se pe ramura în care condiţia este îndeplinită.



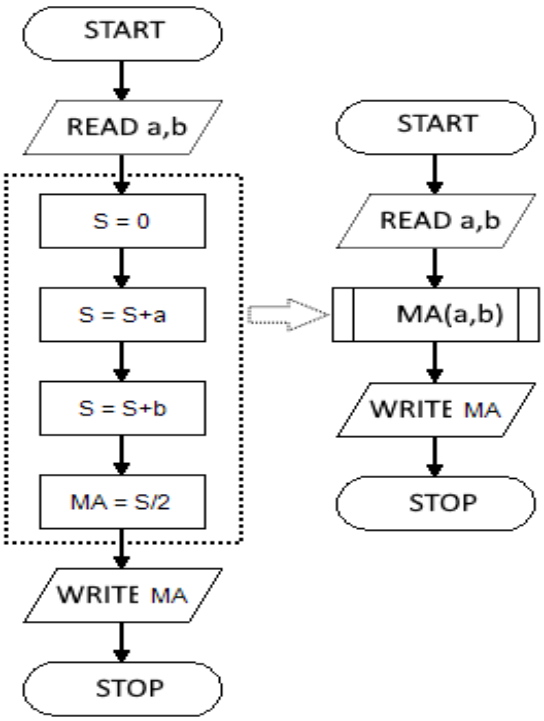
g) procedură-reprezintă un algoritm sau un subalgoritm descris în alt loc, pentru care se cunosc datele de intrare,

Tipuri de structuri

Structura liniară-este dată de un bloc sau o succesiune de blocuri de calcul

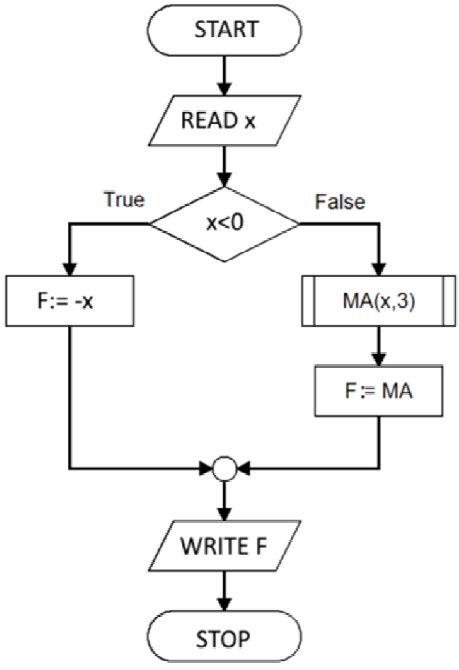


Exemplu: calculul mediei aritmetice a două numere care se citesc. Blocul de calcul al mediei poate fi înlocuit printr-o procedură M(a,b).



Structura alternativă (IF-THEN-ELSE)-este o structură care are la bază blocul de decizie logică. Ca exemplu se consideră schema logica pentru calculul funcţiei:

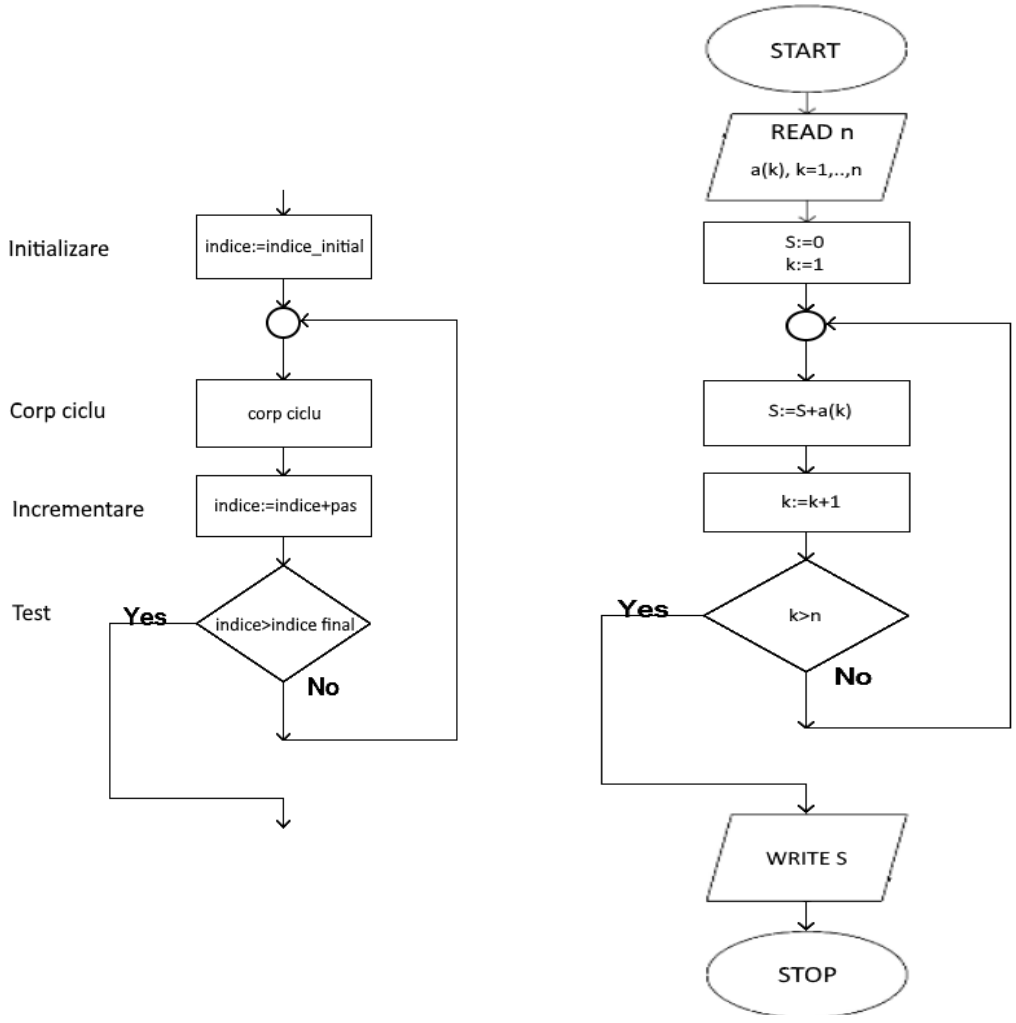
$$F(x) = \begin{cases} \dfrac{x+3}{2}, x \geq 0 \\ -x, x < 0 \end{cases}$$



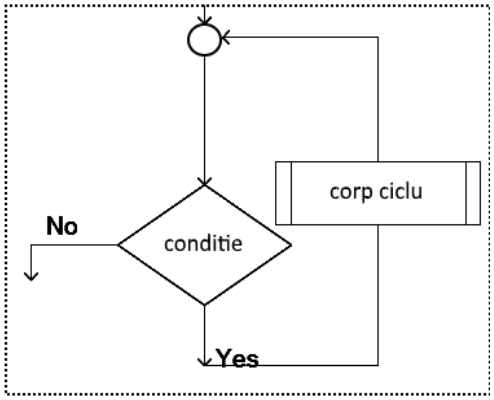
Structuri repetitive (cicluri)-sunt structuri utile în situaţii în care o anumită operaţie sau un grup de operaţii se repetă de un număr finit de on. C. De exemplu calculul sumei elementelor dintr-un şir al, a2, ... , an: S= al+a2+ ... +an .
Operaţiile ce trebuie efectuate pas cu pas sunt:

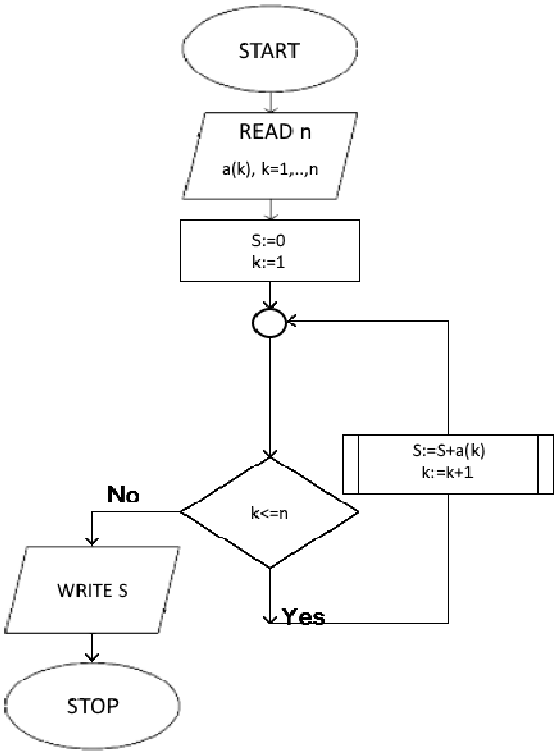
S ← 0 (iniţializarea sumei)
S ←-S+al
S ← S+a2 operaţii care se repetă, diferă indicele
S ←S+an

Ciclu cu contor:

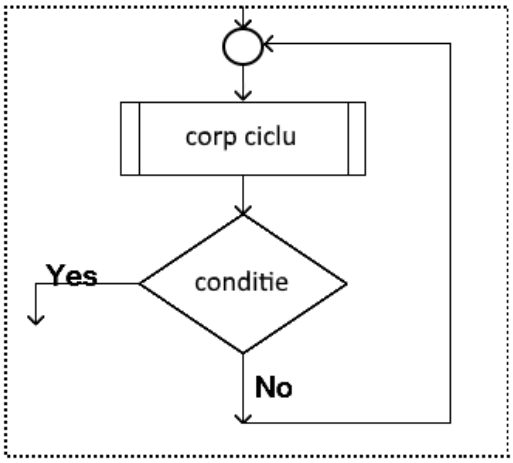


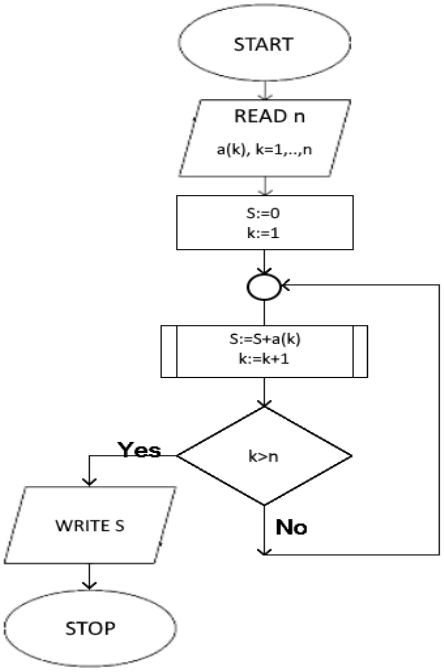
Ciclu cu test iniţial (WHILE-DO) execută corpul ciclului atât timp cât condiţia este adevărată





Ciclu cu test final (REPEAT UNTIL)-Repetă corpul ciclului atât timp cât nu este îndeplinită o condiţie





Structura generală a unui program:

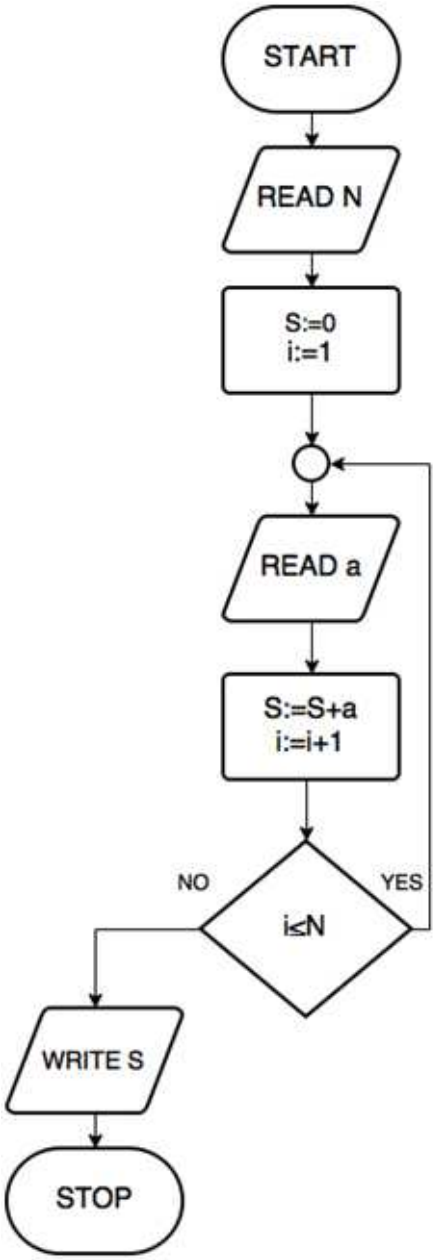
```
program {antetul programului}

uses {zona declaratiilor unitatilor (unituri) de program }
label {zona declaratiilor  etichetelor}
const {zona declaratiilor  constantelor }
type {zona declaratiilor  de tip}
var {zona declaratiilor de variabile }

function {zona declaratiilor  de functii, daca exista}
{ variabile locale }
begin
...
end;

procedure { zona declaratiilor de procedure, daca exista}
{ variabile locale }
begin
...
end;

begin { incepotul zonei programului principal}
...
end. { sfarsitul  programului principal}
```

Pentru calculatoarele pe care se afla instalate versiuni Windows mai recente, se recomanda utilizarea versiunii Pascal ABC.NET care se poate gasi la adresa http://pascalabc.net/en/

3. Probleme rezolvate

1. Sa se scrie un program care sa calculeze suma a N numere reale, pozitive. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logica:

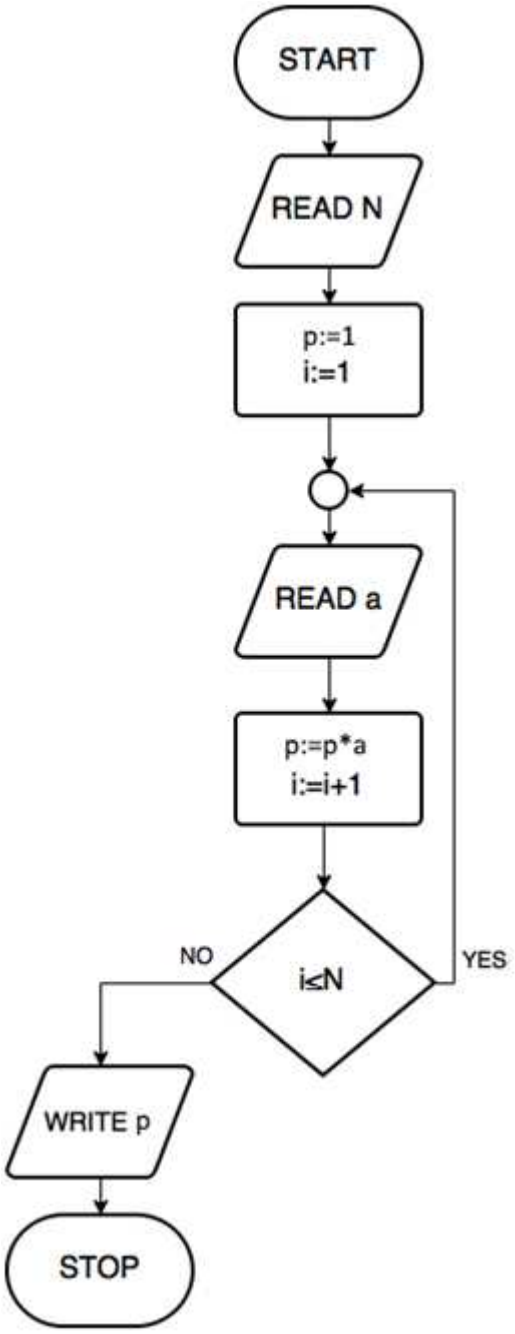


Linii de cod:

```
program suma_n_reale_pozitive;{antet,
poate sa lipseasca}

{blocul de declaratii si definitii:}
var
n,i:byte;
  a,s:real;

{corpul programului:}
begin
write('Introduceti numarul de valori
pozitive N = ');readln(n);

s:=0;{initializare suma}

  {ciclu for pentru introducerea
valorilor de insumat:}
for i:=1 to n do
  begin
    repeat {se verifica daca valorile
introduse de la tastatura sunt
pozitive}
write('A',i,' = ');readln(a);
if a<0 then writeln('A este negativ,
introduceti o valoare pozitiva !');
until a>=0;{sfarsit verificare}
s:=s+a
end;
writeln('Suma = ',s:8:2);{afisare
suma}

readln; );{mentinere ecran
utilizator}

end.
```

2. Sa se scrie un program care sa calculeze produsul a N numere reale, pozitive. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

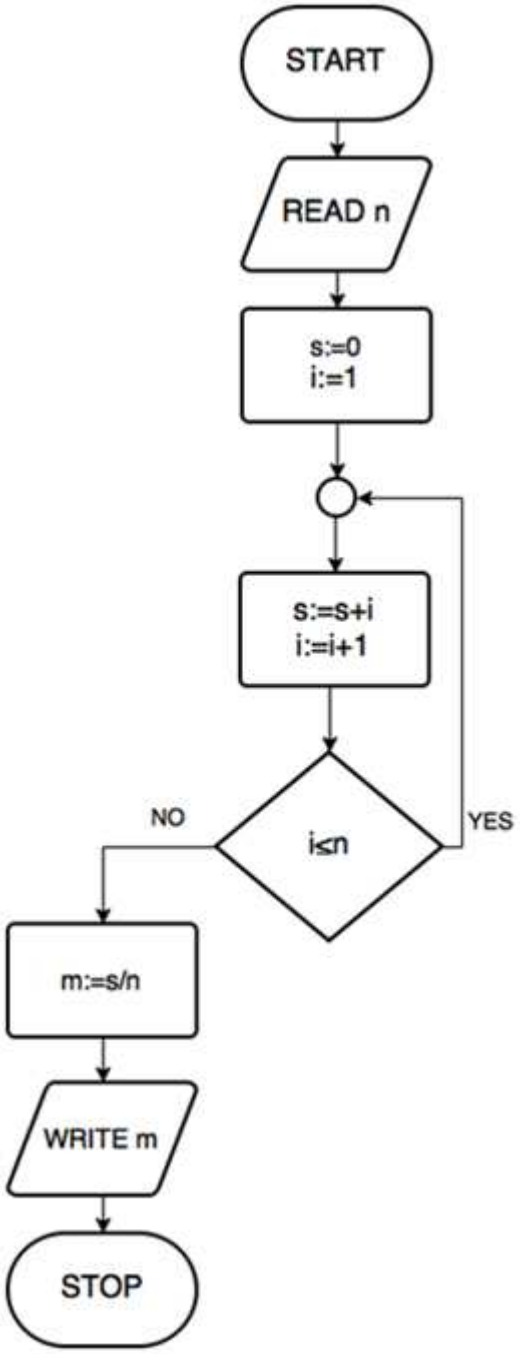


Linii de cod:

```
program
produs_n_reale_pozitive;{antet}

{blocul de declaratii si
definitii:}
var
n,i:byte;
  a,p:real;

{corpul programului:}
begin
write('Introduceti numarul de
valori pozitive N = ');readln(n);

p:=1;{initializare produs}

  {ciclu for pentru introducerea
valorilor insumate:}
for i:=1 to n do
  begin
    repeat {se verifica daca
valorile introduse de la tastatura
sunt pozitive}
write('A',i,' = ');readln(a);
if a<0 then writeln('A este
negativ, introduceti o valoare
pozitiva !');
until a>=0;{sfarsit verificare}
p:=p*a
end;
writeln('Suma = ',p:8:2);{afisare
suma}
readln;
end.
```

3. Sa se scrie un program care sa calculeze media a N numere naturale. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logica:

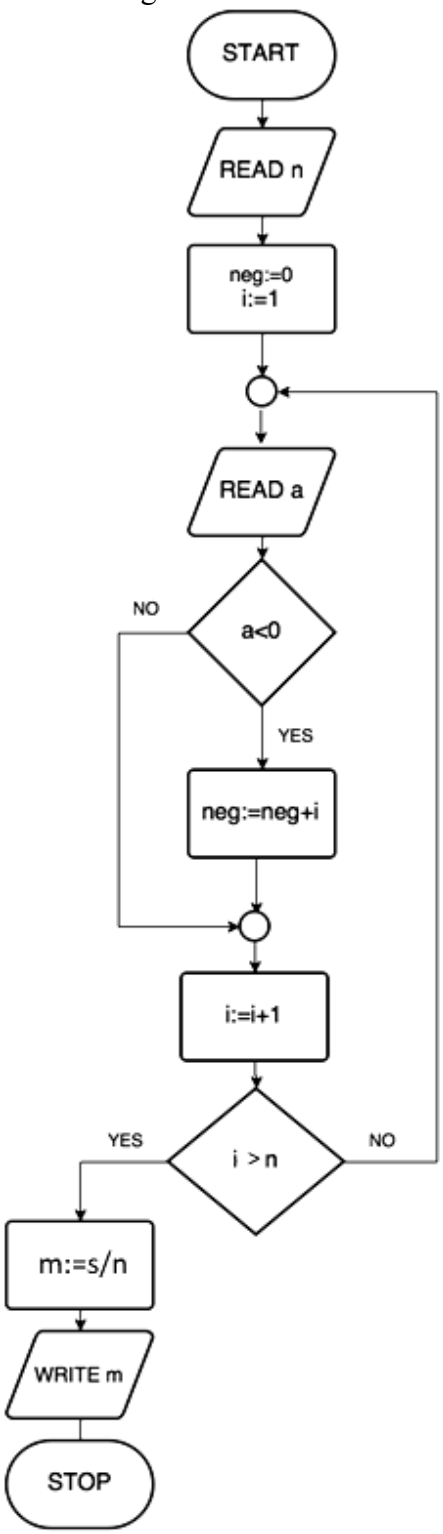


Linii de cod:

```
program media_nr_nat;{antet}

{blocul de declaratii si
definitii:}
var
n,i,s:integer;
  m:real;

{corpul programului:}
begin
write('Introduceti numarul de
numere naturale N =
');readln(n);

s:=0;{initializare suma}

  {ciclu for pentru
introducerea valorilor
insumate:}
for i:=1 to n do
s:=s+i;

m:=s/n;
writeln('media =
',m:8:2);{afisare suma}
readln;
end.
```

4. Sa se scrie un program care sa calculeze numărul de valori negative dintr-un total de N numere reale introduse de la tastatură. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

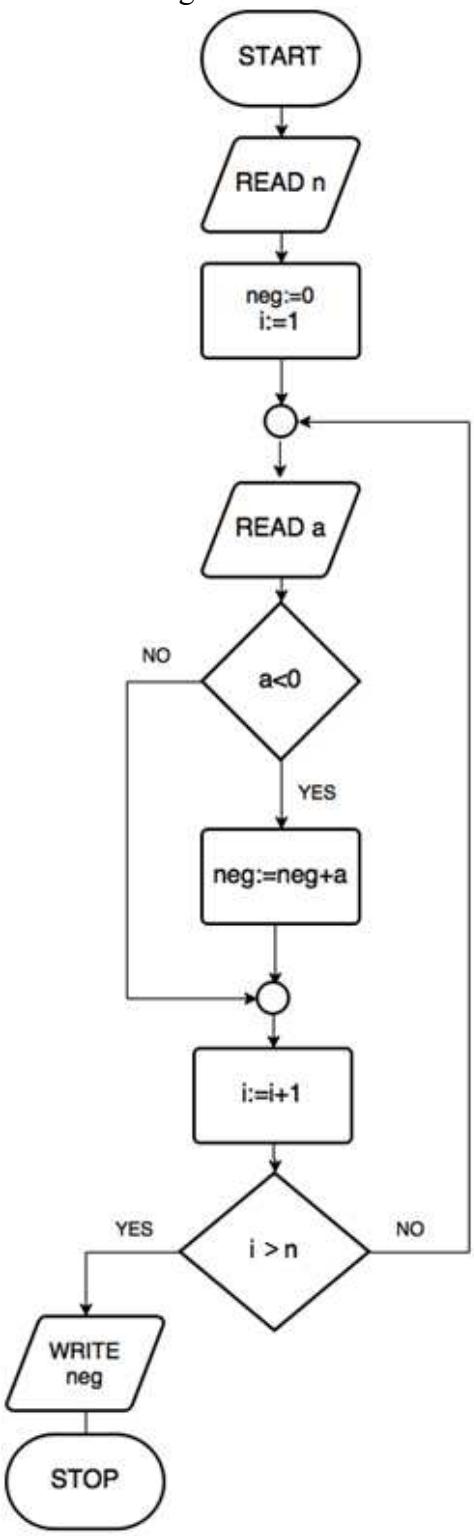


Linii de cod:

```
program numar_val_negative;{antet}

{blocul de declaratii si
definitii:}
var
n,i,neg:byte;
  a:real;

{corpul programului:}
begin
write('Introduceti numarul total de
numere reale analizate N = ');
readln(n);

neg:=0;{initializare}

  {ciclu for pentru introducerea
valorilor analizate:}
for i:=1 to n do
  begin
write('A',i,' = ');readln(a);
if a<0 then {verificare nr
negative}
neg:=neg+1 {incrementare total nr
negative}
end;
writeln('Total negative =
',neg);{afisare total numere
negative}
readln;
end.
```

5. Sa se scrie un program care sa calculeze suma valorilor negative dintr-un şir de N numere reale introduse de la tastatura. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

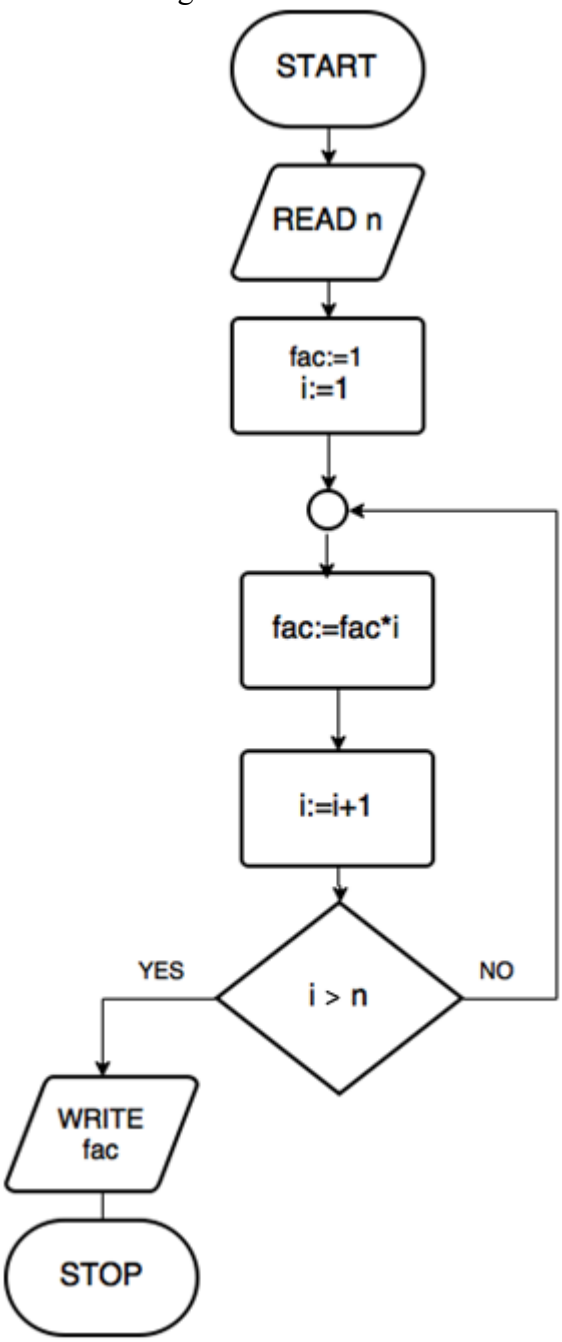


Linii de cod:

```
program suma_val_negative;{antet}

{blocul de declaratii si definitii:}
var
n,i:byte;
  a,neg:real;

{corpul programului:}
begin
write('Introduceti numarul total de
numere reale analizate N = ');
readln(n);

neg:=0;{initializare}

  {ciclu for pentru introducerea
valorilor analizate:}
for i:=1 to n do
  begin
write('A',i,' = ');readln(a);
if a<0 then {verificare nr negative}
neg:=neg+a{suma a negative}
end;
writeln('Suma negative =
',neg);{afisare total numere
negative}
readln;
     end.
```

6. Sa se scrie un program care sa calculeze N!. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

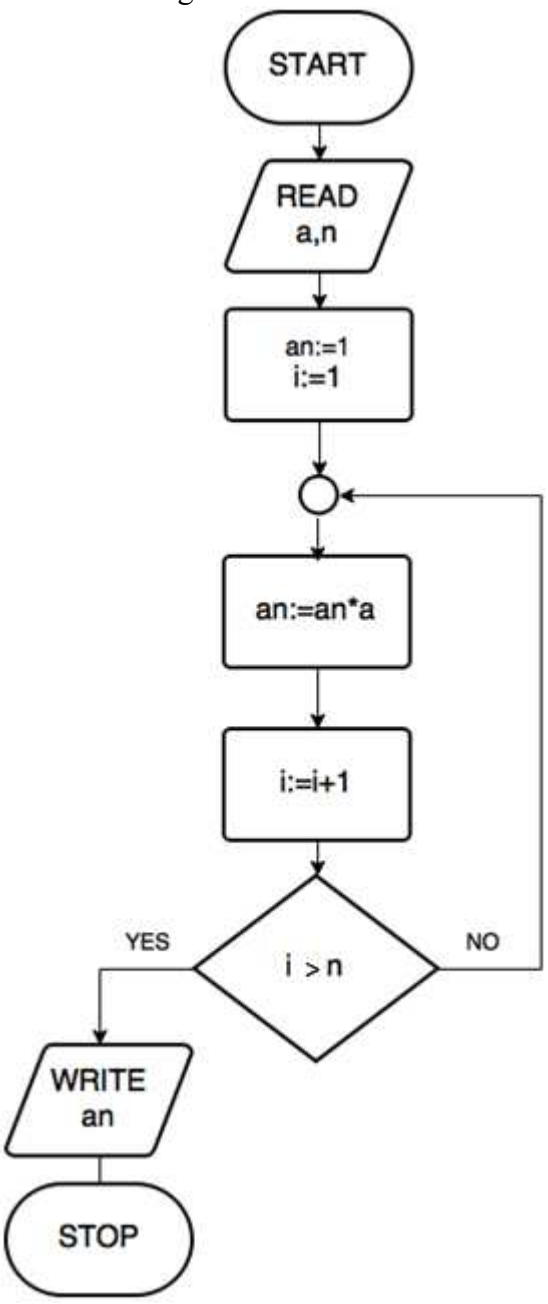


Linii de cod:

```
program n_factorial;{antet}

{blocul de declaratii si
definitii:}
var
n,i,fac:integer;

{corpul programului:}
begin
write('Introduceti  N =
');readln(n);

fac:=1;{initializare}

  {ciclu for pentru calculare
factorial:}
for i:=1 to n do
fac:=fac*i;

writeln('N factorial =
',fac);{afisare total numere
negative}

readln;
end.
```

7. Sa se scrie un program care sa calculeze A^N, A este variabilă reală, N întreg, pozitivă. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

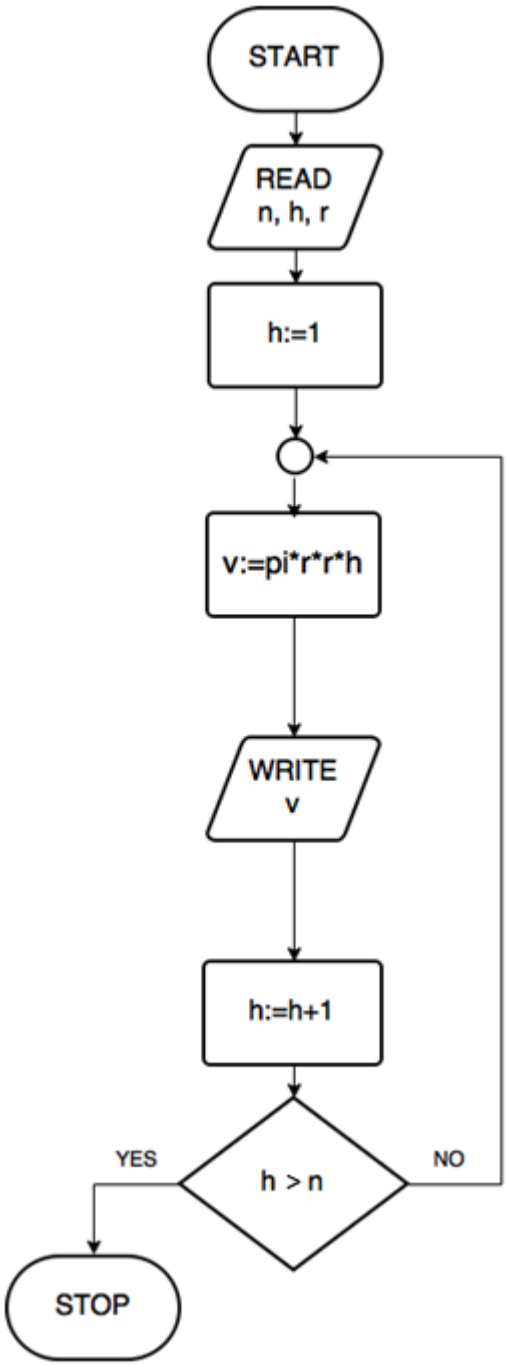


Linii de cod:

```
program a_la_n;{antet}

{blocul de declaratii si
definitii:}
var
n,i,a,an:integer;

{corpul programului:}
begin
{date intrare:}
write('Introduceti  A =
');readln(a);
write('Introduceti  N =
');readln(n);

an:=1;{initializare}

  {ciclu for pentru calculare
putere:}
for i:=1 to n do
an:=an*a;

writeln(a,' la ',n,' =
',an);{afisare a la n}

readln;
end.
```

8. Sa se scrie un program care sa calculeze volumul unui cilindru de rază R (real) şi generatoarea H cu valori întregi cuprinse intre 1 şi N. Toate cererile şi rezultatele vor fi afişate explicit pe ecran .

Schema logică:

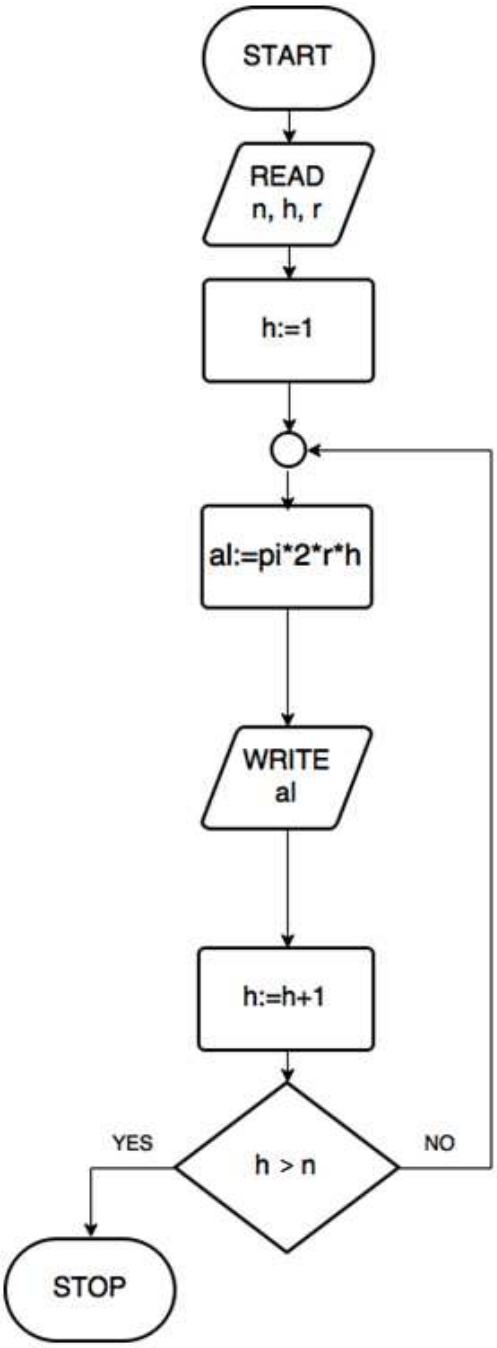


Linii de cod:

```
program vol_cil;{antet}

{blocul de declaratii si definitii:}
var
n,h:byte;
  r, v:real;


{corpul programului:}
begin
{date intrare:}
write('Introduceti  R = ');
readln(r);
write('Introduceti  N = ');
readln(n);

{ciclu cu instructiune compusa
pentru calculare-afisare volum:}
for h:=1 to n do
  begin
v:=pi*r*r*h;
  writeln('la h = ',h,',  volumul
este v = ',v:6:2,';');{afisare
volum}
end;
readln;
end.
```

9. Sa se scrie un program care sa calculeze aria laterală a unui cilindru drept de rază R (real) şi generatoarea H cu valori intregi cuprinse intre 1 şi N. Toate cererile şi rezultatele vor fi afişate explicit pe ecran .

Schema logică:

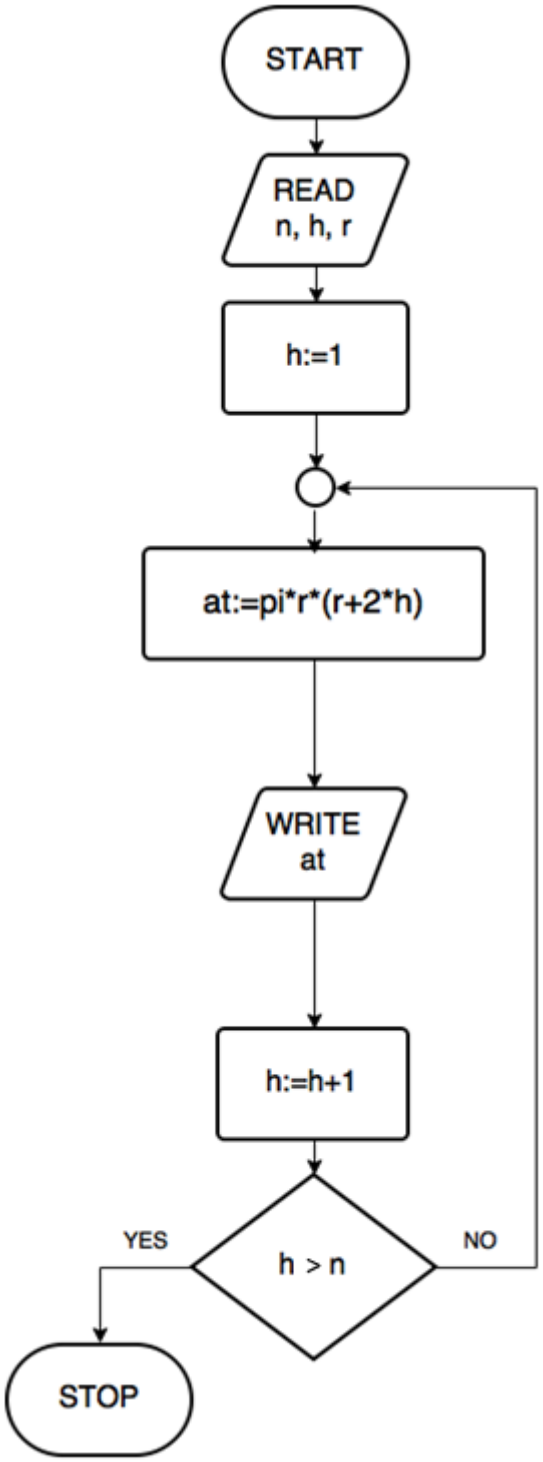


Linii de cod:

```
program a_l_cil;{antet}

{blocul de declaratii si
definitii:}
var
n,h:byte;
  r, al:real;


{corpul programului:}
begin
{date intrare:}
write('Introduceti  R = ');
readln(r);
write('Introduceti  N = ');
readln(n);

{ciclu cu instructiune compusa
pentru calculare-afisare al:}
for h:=1 to n do
  begin
al:=pi*2*r*h;
  writeln('la h = ',h,',  aria
laterala este al = ',al:6:2,';');
end;
readln;
end.
```

10. Sa se scrie un program care sa calculeze aria totală a unui cilindru drept de rază R (real) şi generatoarea H cu valori intregi cuprinse intre 1 şi N. Toate cererile şi rezultatele vor fi afişate explicit pe ecran .

Schema logică:

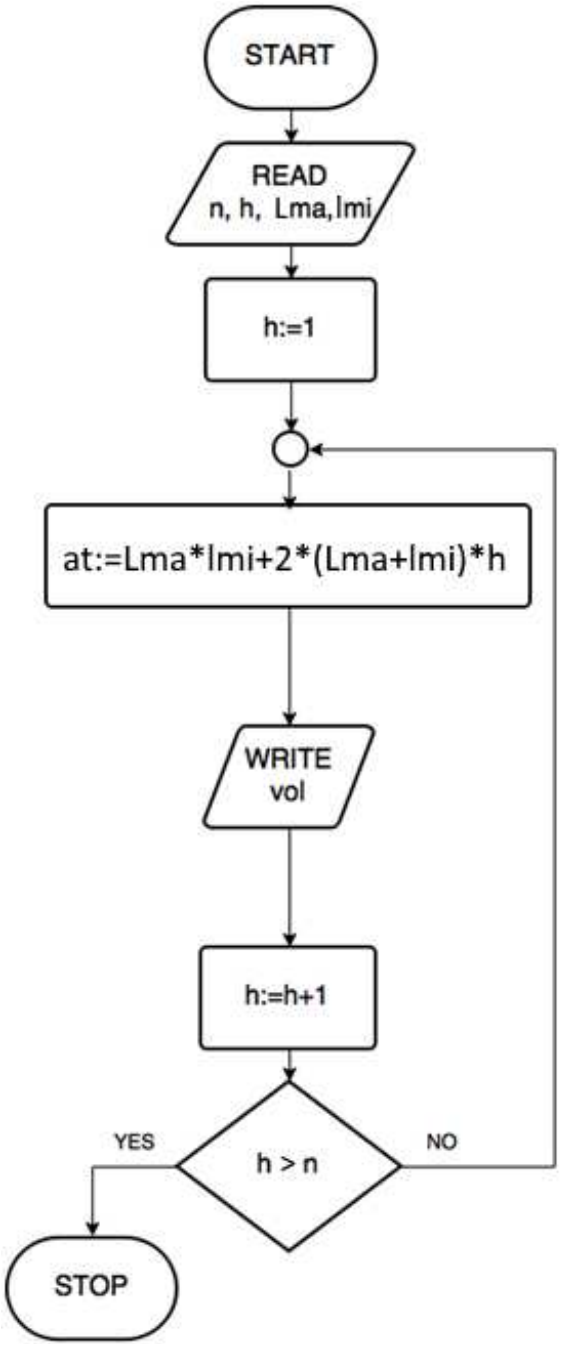


Linii de cod:

```
program a_tot_cil;{antet}

{blocul de declaratii si
definitii:}
var
n,h:byte;
  r, at:real;


{corpul programului:}
begin
{date intrare:}
write('Introduceti  R = ');
readln(r);
write('Introduceti  N = ');
readln(n);

{ciclu cu instructiune compusa
pentru calculare-afisare at:}
for h:=1 to n do
  begin
at:=pi*r*(r+2*h);
  writeln('la h = ',h,', aria
totala este at = ',al:6:2,';');
end;
readln;
end.
```

11. Sa se scrie un program care sa calculeze aria totală a unei prisme drepte cu baza un dreptunghi având laturile L şi l (întregi) şi înălţimea H cu valori intregi cuprinse intre 1 şi N. Toate cererile şi rezultatele vor fi afişate explicit pe ecran .

Schema logică:

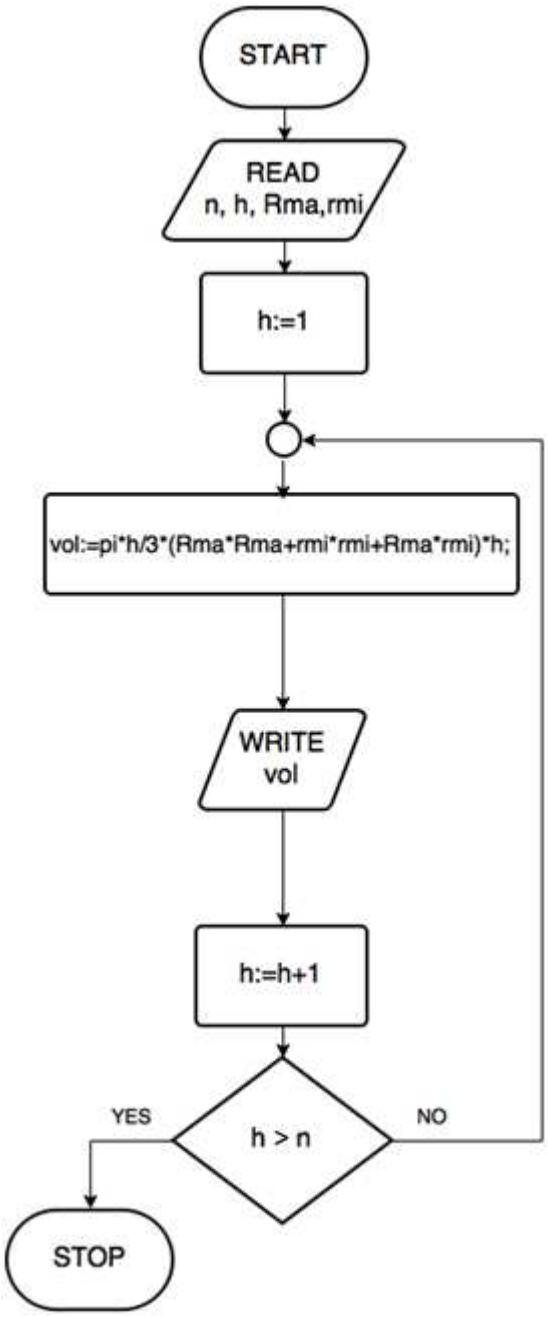


Linii de cod:

```
program a_tot_prisma;{antet}

{blocul de declaratii si
definitii:}
var
n,h:byte;
  Lma, lmi, at:real;


{corpul programului:}
begin
{date intrare:}
write('Introduceti  Lma = ');
readln(Lma);
write('Introduceti  lmi = ');
readln(lmi);
write('Introduceti  N = ');
readln(n);

{ciclu cu instructiune compusa
pentru calculare-afisare at:}
for h:=1 to n do
  begin
at:=Lma*lmi+2*(Lma+lmi)*h;
  writeln('la h = ',h,', aria
totala este at = ',at:6:2,';');
end;
readln;
end.
```

12. Sa se scrie un program care să calculeze volumul unui trunchi de con având raza bazei mari R, respectiv raza bazei mici r şi înălţimea H numere întregi pozitive. Toate cererile şi rezultatele vor fi afişate explicit pe ecran . ($v=\pi h/3(R^2+r^2+Rr)$)

Schema logică:

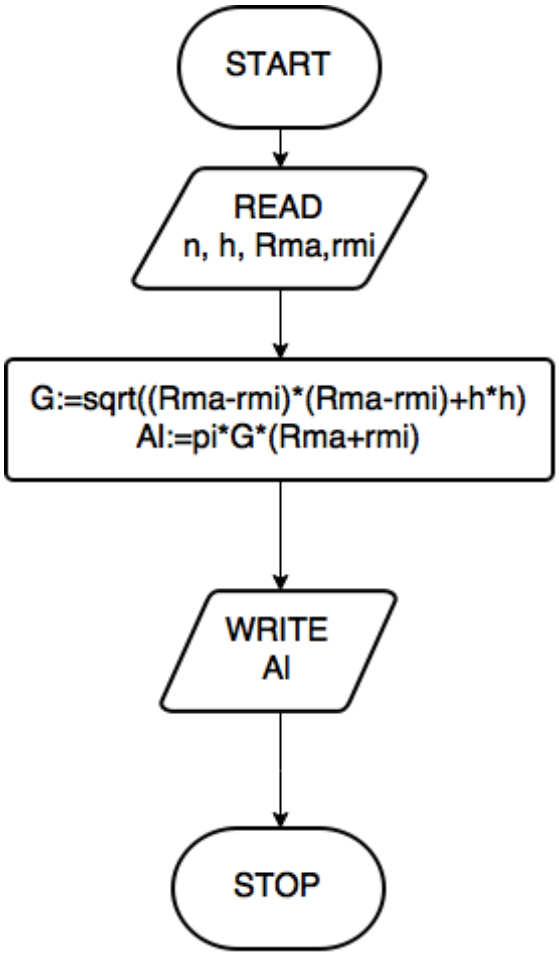


Linii de cod:

```
program vol_tr_con;{antet}

{blocul de declaratii si
definitii:}
var
n,h:byte;
  Rma, rmi, vol:real;


{corpul programului:}
begin
{date intrare:}
write('Introduceti  Rma = ');
readln(Rma);
write('Introduceti  rmi = ');
readln(rmi);
write('Introduceti  N = ');
readln(n);

{ciclu cu instructiune compusa
pentru calculare-afisare at:}
for h:=1 to n do
  begin
vol:=pi*h/3*(Rma*Rma+rmi*rmi+Rma*r
mi)*h;
  writeln('la h = ',h,', volumul
este vol = ',vol:6:2,';');
{afisare vol}
end;
readln;
```

13. Sa se scrie un program care sa calculeze aria laterală ale unui trunchi de con având raza bazei mari R, respectiv raza bazei mici r şi înălţimea H numere întregi pozitive. Toate cererile şi rezultatele vor fi afişate explicit pe ecran . (Al=πG(R+r)

Schema logică:

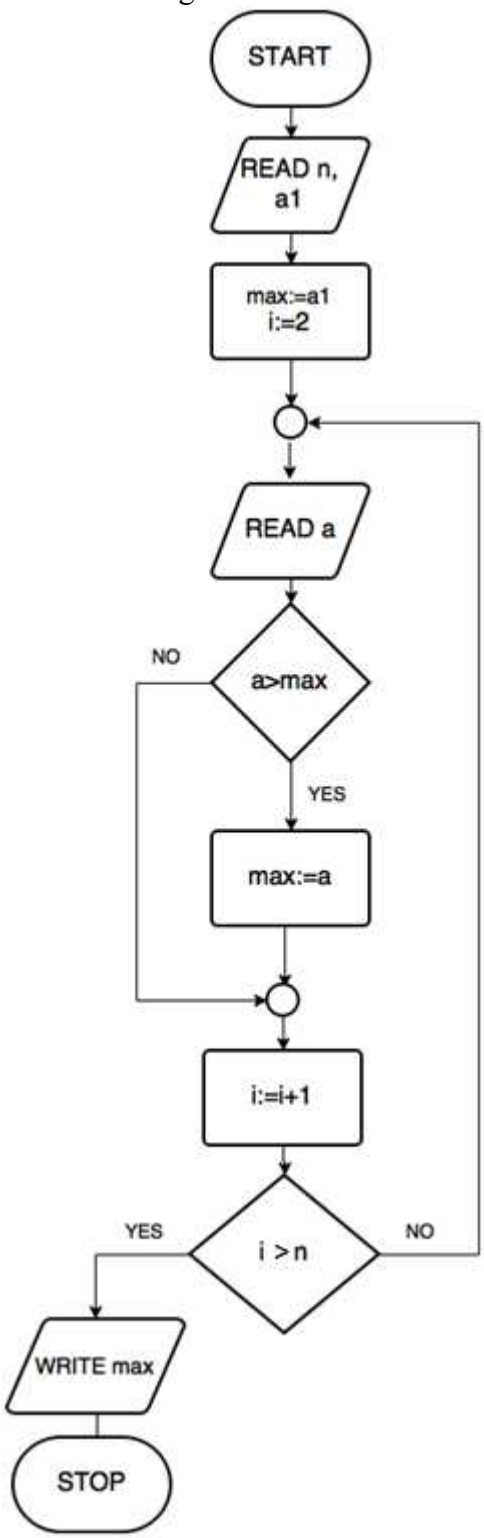


Linii de cod:

```
program al_tr_con;{antet}

{blocul de declaratii si
definitii:}
var
n,h:byte;
  Rma, rmi, al:real;

{corpul programului:}
begin
{date intrare:}
write('Introduceti  Rma = ');
readln(Rma);
write('Introduceti  rmi = ');
readln(rmi);
write('Introduceti  H = ');
readln(h);

  G:=sqrt((Rma-rmi)*(Rma-
rmi)+h*h);
  Al:=pi*G*(Rma+rmi);
  writeln('Aria laterala = ',
Al:6:2,';');
{afisare Al}

readln;
end.
```

14. Sa se scrie un program care sa calculeze valoarea maximă dintr-un şir de N numere reale. Toate cererile şi rezultatele vor fi afişate explicit pe ecran .

Schema logică:



Linii de cod:

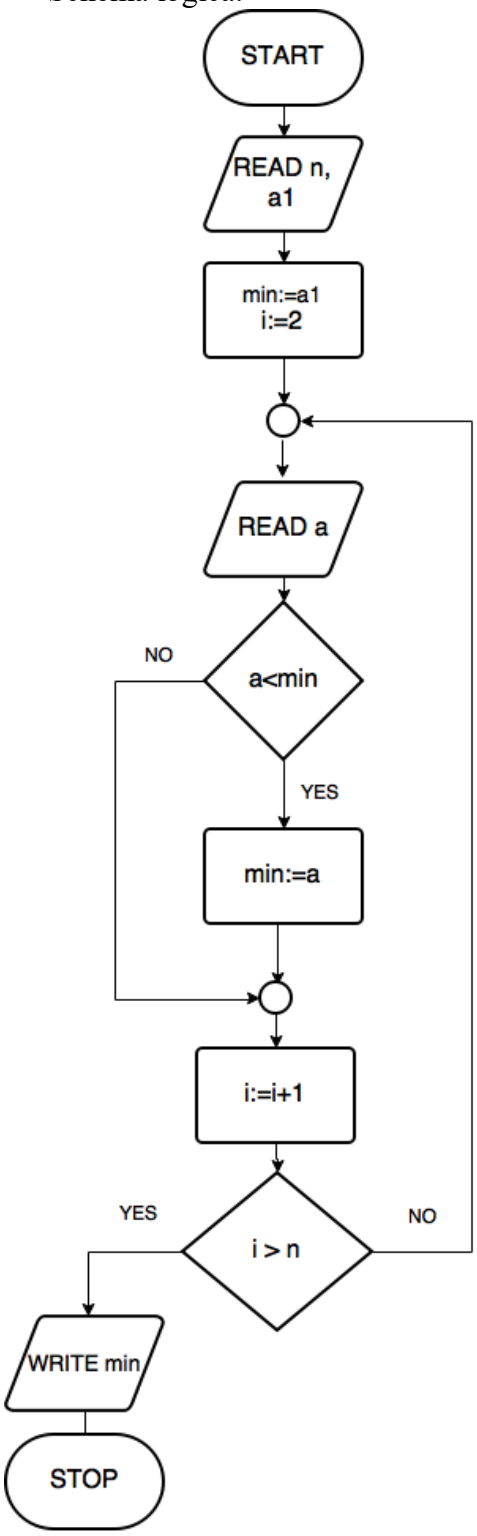
```
program max_real;{antet}

{blocul de declaratii si definitii:}
var
n,i:byte;
  a,max:real;

{corpul programului:}
begin
write('Introduceti numarul total de
numere reale analizate N = ');
  readln(n);
  write('A1 = ');readln(a);
  max:=a;{initializare}

  {ciclu for pentru introducerea
valorilor analizate si comparare:}
for i:=2to n do
  begin
write('A',i,' = ');readln(a);
if max<a then {verificare maxim}
max:=a;
end;

{afisare maxim:}
  writeln('Valoare maxima = ',max);

readln;
end.
```

Varianta de rezolvare utilizand o variabilă de tip tablou pentru stocarea datelor introduse de la tastatură:

```
program max_real_tab;{antet}

{blocul de declaratii si definitii:}
var
n,i:byte;
  a:array[1..100] of real;
  max:real;

{corpul programului:}
begin

write('Introduceti numarul total de numere reale analizate
N = ');
  readln(n);

{Introducere primei valori analizate:}
write('A1 = ');readln(a[1]);

{initializare maxim la prima valoare:}
max:=a[1];

{ciclu for pentru introducerea valorilor analizate:}
     for i:=2 to n do
       begin
     write('A',i,' = ');readln(a[i]);
     end;

{calculare maxim:}
     for i:=1 to n do
       if max<a[i] then max:=a[i];

{afisare maxim:}
writeln('Valoare maxima = ',max);

  readln;
end.
```

15. Sa se scrie un program care să calculeze valoarea minimă dintr-un şir de N numere reale. Toate cererile şi rezultatele vor fi afişate explicit pe ecran .

Schema logică:



Linii de cod:

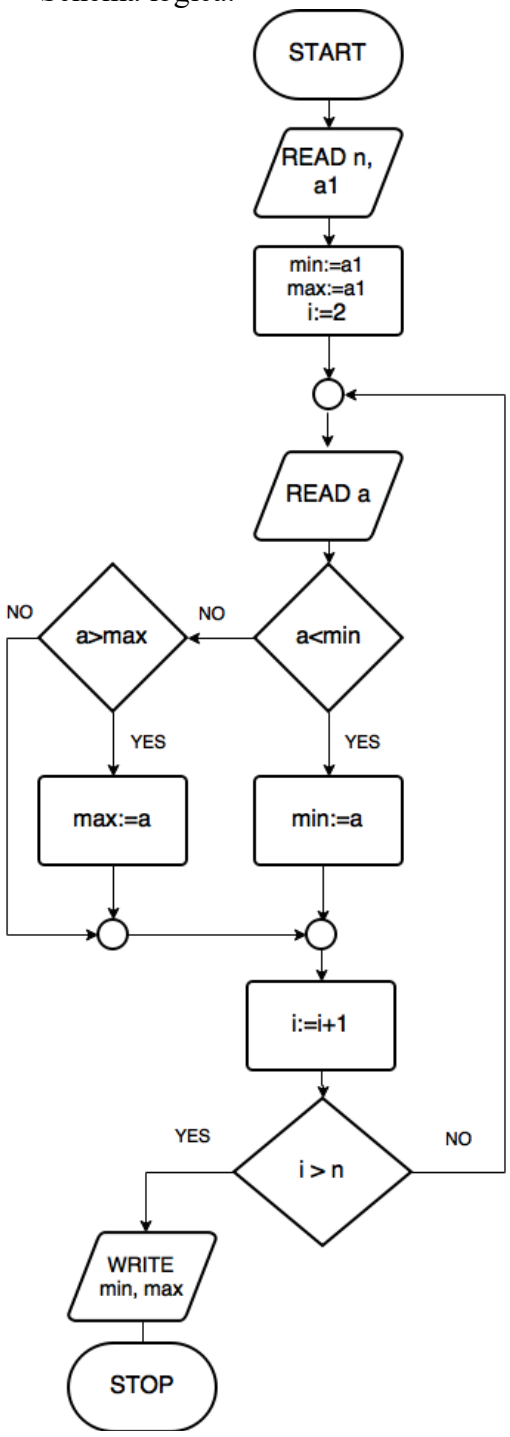
```
program min_real;{antet}

{blocul de declaratii si definitii:}
var
n,i:byte;
  a,min:real;

{corpul programului:}
begin
write('Introduceti numarul total de
numere reale analizate N = ');
  readln(n);
  write('A1 = ');readln(a);
  min:=a;{initializare}

  {ciclu for pentru introducerea
valorilor analizate si comparare:}
for i:=2to n do
  begin
write('A',i,' = ');readln(a);
if min>a then {verificare minim}
min:=a;
end;

{afisare minim:}
  writeln('Valoare minima = ',min);

readln;
end.
```

Varianta de rezolvare utilizand o variabilă de tip tablou pentru stocarea datelor introduse de la tastatură:

```
program min_real_tab;{antet}

{blocul de declaratii si definitii:}
var
n,i:byte;
  a:array[1..100] of real;
  min:real;

{corpul programului:}
begin

write('Introduceti numarul total de numere reale analizate
N = ');
  readln(n);

{Introducerea primei valori analizate:}
write('A1 = ');readln(a[1]);

{initializare minim la prima valoare:}
min:=a[1];

{ciclu for pentru introducerea valorilor analizate:}
     for i:=2 to n do
       begin
     write('A',i,' = ');readln(a[i]);
     end;

{calculare minim:}
     for i:=1 to n do
       if min>a[i] then min:=a[i];

{afisare minim:}
writeln('Valoare minima = ',min);

  readln;
end.
```

16. Sa se scrie un program care sa calculeze simultan valoarea maximă şi minimă dintr-un şir de N numere reale. Toate cererile şi rezultatele vor fi afişate explicit pe ecran .

Schema logică:



Linii de cod:

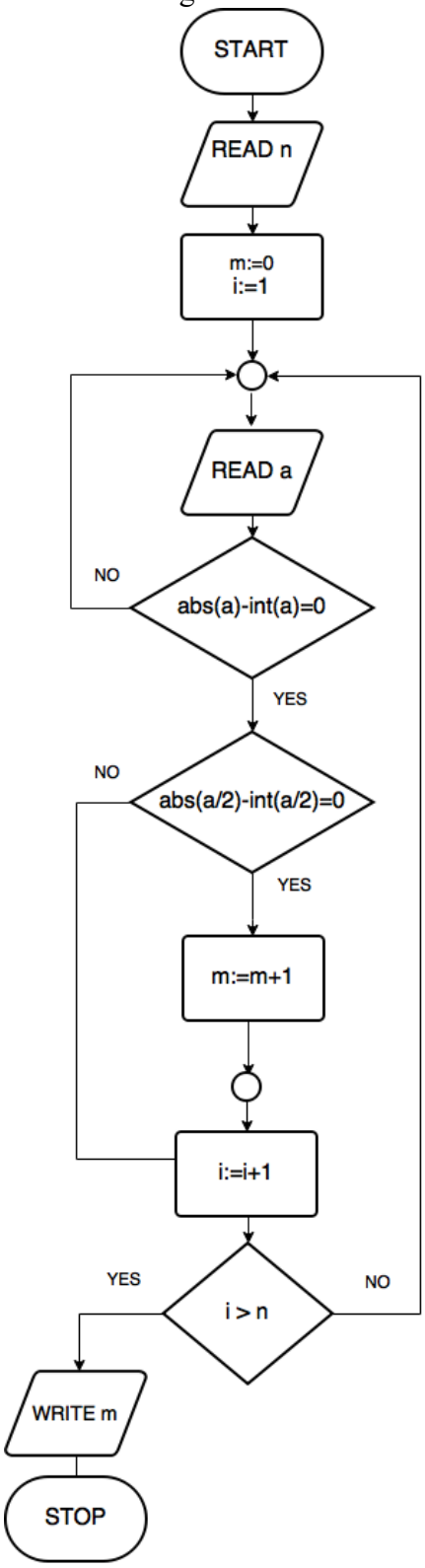
```
program max_min_real;{antet}

{blocul de declaratii si definitii:}
var
n,i:byte;
  a,max, min:real;

{corpul programului:}
begin
write('Introduceti numarul total de
numere reale analizate N = ');
  readln(n);
  write('A1 = ');readln(a);
  max:=a; min:=a;{initializare}

  {ciclu for pentru introducerea
valorilor analizate si comparare:}
for i:=2 to n do
  begin
write('A',i,' = ');readln(a);
if max<a then {verificare maxim}
max:=a;
if min>a then {verificare minim}
min:=a;
end;

{afisare minim:}
writeln('Valoare maxima = ',max);
  writeln('Valoare minima = ',min);

  readln;
end.
```

Varianta de rezolvare utilizand o variabilă de tip tablou pentru stocarea datelor introduse de la tastatură:

```
program min_max_real_tab;{antet}

{blocul de declaratii si definitii:}
var
n,i:byte;
  a:array[1..100] of real;
  min, max:real;

{corpul programului:}
begin

write('Introduceti numarul de numere reale analizate N = ');
  readln(n);

{Introducerea primei valori analizate:}
write('A1 = ');readln(a[1]);

{initializare minim si maxim la prima valoare:}
min:=a[1];
  max:=a[1];

{ciclu for pentru introducerea valorilor analizate:}
for i:=2 to n do
  begin
write('A',i,' = ');readln(a[i]);
end;

{calculare minim si maxim:}
for i:=1 to n do
  begin
   if min>a[i] then min:=a[i];
if max<a[i] then max:=a[i];
end;

{afisare minim si maxim:}
writeln('Valoare minima = ',min);
  writeln('Valoare maxima = ',max);

  readln;
end.
```

17. Sa se scrie un program care sa calculeze numarul valorilor pare dintr-un şir de N numere naturale introduse de la tastatura. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:



Linii de cod:

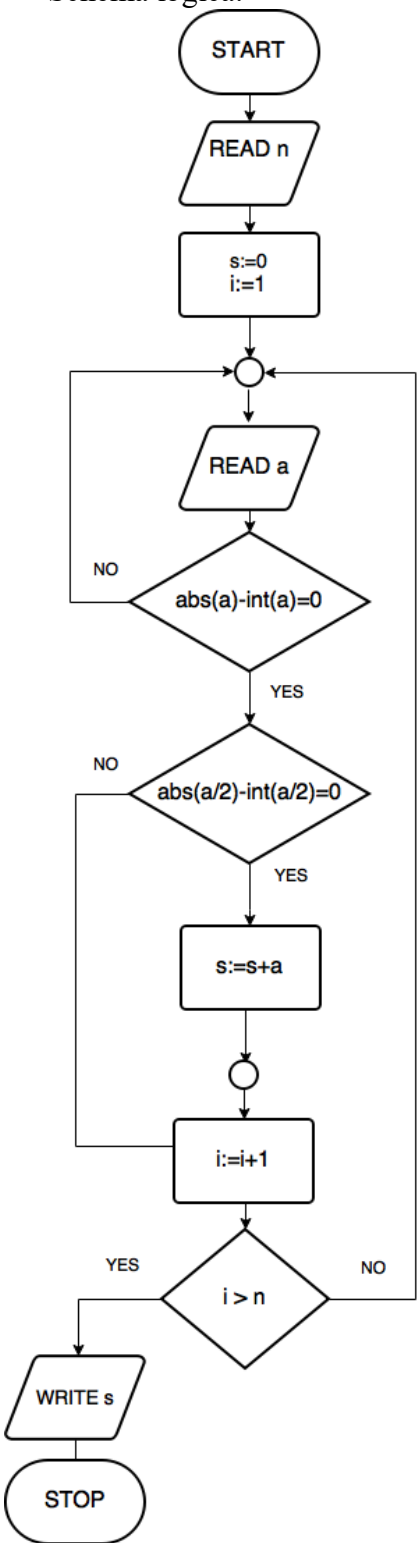
```
program nat_pare;{antet}

{blocul de declaratii si definitii:}
var
n,i,m:word;
  a:real;

{corpul programului:}
begin

write('Introduceti numarul de numere
naturale analizate N = ');
  readln(n);

  m:=0; {initializare variabila total
numere pare}
  {ciclu for pentru introducerea
valorilor analizate:}
for i:=1 to n do
    repeat {verificare ca datele
introduse sunt intregi pozitive}
write('A',i,' = ');readln(a);
if abs(a/2)-int(a/2)=0 {restul
impartiriila 2 =0}
then{incrementare m, numere pare}
m:=m+1
until abs(a)-int(a)=0;

{afisare total numere pare:}
writeln('Total numere pare m = ',m);

  readln;
end.
```

Variantă de program în care restricționarea la numere naturale se face prin definirea variabilei care stocheaza aceste numere ca *variabila de tipword*. În acest caz introducerea unei valori din afara domeniului va determina oprirea programului cu semnalarea unei erori.

```
program nat_pare;{antet}

{blocul de declaratii si definitii:}
var
n,i,m,a:word;

{corpul programului:}
begin

write('Numarul de numere naturale analizate N = ');
  readln(n);

  m:=0; {initializare variabila total numere pare}
  {ciclu for pentru introducerea valorilor analizate:}
for i:=1 to n do
 begin
write('A',i,' = ');readln(a);
if a mod 2 = 0 then
m:=m+1 {incrementare m, numere pare}
end;

{afisare total numere pare:}
writeln('Total numere pare m = ',m);

  readln;
end.
```

18. Sa se scrie un program care sa calculeze suma valorilor pare dintr-un şir de N numere naturale introduse de la tastatura. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:



Linii de cod:

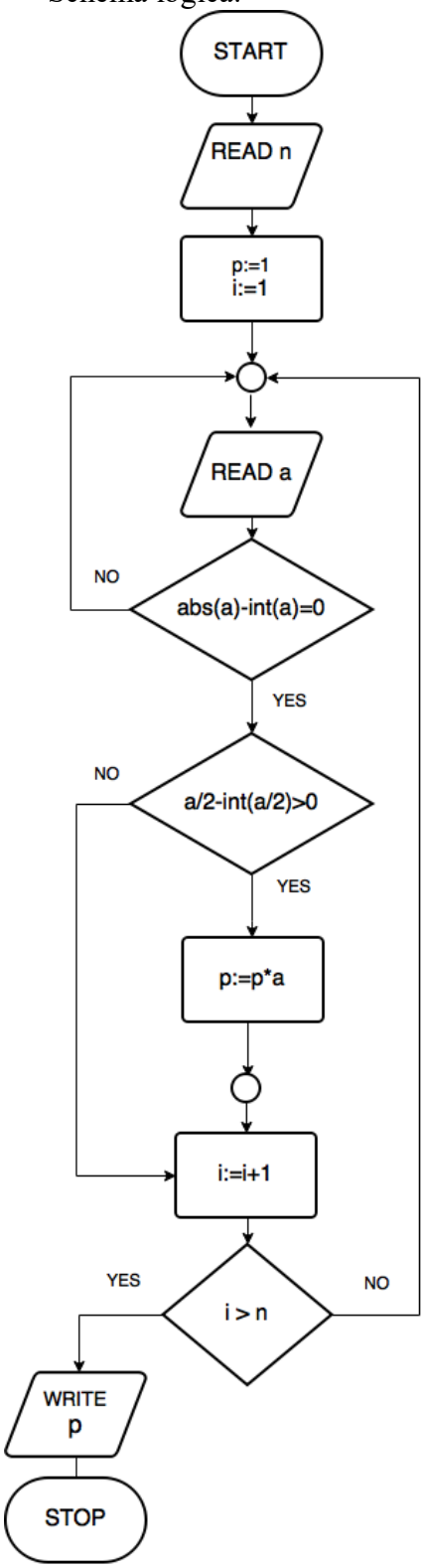
```
program suma_pare;{antet}

{blocul de declaratii si definitii:}
var
n,i:word;
  a,s:real;

{corpul programului:}
begin

write('Introduceti numarul de numere
naturale analizate N = ');
  readln(n);

  s:=0; {initializare variabila total
numere pare}
  {ciclu for pentru introducerea
valorilor analizate:}
for i:=1 to n do
    repeat {verificare ca datele
introduse sunt intregi pozitive}
write('A',i,' = ');readln(a);
if abs(a/2)-int(a/2)=0 {restul
impartiriila 2 =0}
then{incrementare m, numere pare}
s:=s+a
until abs(a)-int(a)=0;

{afisare total numere pare:}
writeln('Suma numere pare S = ',s);

  readln;
end.
```

Variantă de program în care restricționarea la numere naturale se face prin definirea unei variabile tablou cu tipul*word*. În acest caz introducerea unei valori din afara domeniului va determina oprirea programului cu semnalarea unei erori.Secventa de introducere a datelor este separată de secventa de procesare.

```
program suma_pare;{antet}

{blocul de declaratii si definitii:}
var
n,i,s:word;
  a:array[1..100] of word;

{corpul programului:}
begin
write('Numarul de numere naturale analizate N = ');
  readln(n);

{ciclu for pentru introducerea valorilor analizate:}
for i:=1 to n do
begin
write('A',i,' = ');readln(a[i]);
end;

s:=0; {initializare variabila suma numere pare}

for i:=1 to n do
if a[i]mod 2 = 0 then
s:=s+a[i];{suma numere pare}

{afisare total numere pare:}
writeln('Suma numere pare S = ',s);

  readln;
end.
```

19. Sa se scrie un program care sa calculeze produsul valorilor impare dintr-un şir de N numere naturale introduse de la tastatura. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:



Linii de cod:

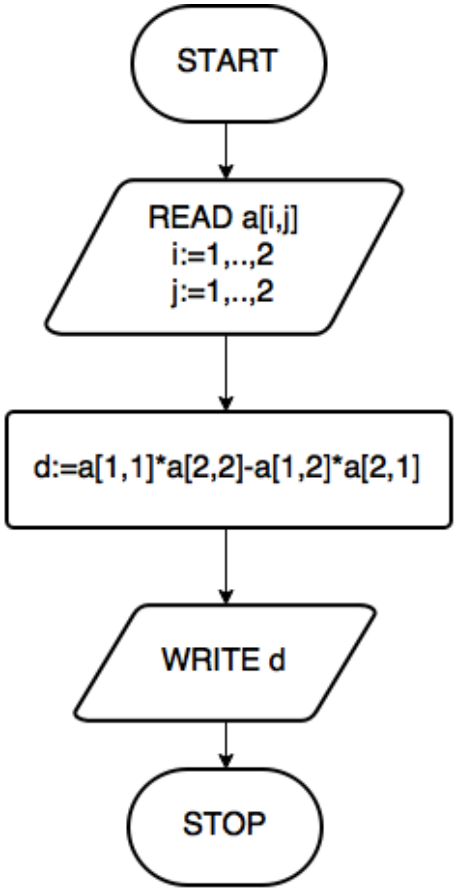
```
program produs_impare;{antet}

{blocul de declaratii si definitii:}
var
n,i:word;
  a,p:real;

{corpul programului:}
begin

write('Numarul de numere naturale de
analizat N = ');readln(n);

{initializare variabila produs numere
impare:}
  p:=1;

{ciclu for pentru introducerea
valorilor analizate:}
for i:=1 to n do
begin
{verificare ca datele introduse sunt
intregi pozitive:}
repeat
write('A',i,' = ');readln(a);
until abs(a)-int(a)=0;

{se verifica daca restul impartiriila
2 > 0:}
if a/2-int(a/2)>0 then  p:=p*a
end;

{afisare produs numere impare:}
writeln('Produs numere impare P =
',p);

  readln;
end.
```

Variantă de program în care restricționarea la numere naturale se face prin definirea unei variabile tablou cu tipul*word*. În acest caz introducerea unei valori din afara domeniului va determina oprirea programului cu semnalarea unei erori. Secventa de introducere a datelor este separată de secventa de procesare.

```
program produs_impare;{antet}

{blocul de declaratii si definitii:}
var
n,i,p:word;
  a:array[1..100] of word;

{corpul programului:}
begin
write('Numarul de numere naturale analizate N = ');
  readln(n);

{ciclu for pentru introducerea valorilor analizate:}
for i:=1 to n do
begin
write('A',i,' = ');readln(a[i]);
end;

p:=1; {initializare variabila produs numere impare}

for i:=1 to n do
if a[i]mod 2 <>0 then
p:=p*a[i];{produs numere impare}

{afisare rezultat:}
writeln('Produs numere impare P = ',p);

  readln;
end.
```

20. Sa se scrie un program care sa calculeze valoarea unui determinant de ordinul 2 (D=a11*a22-a12*a21). Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

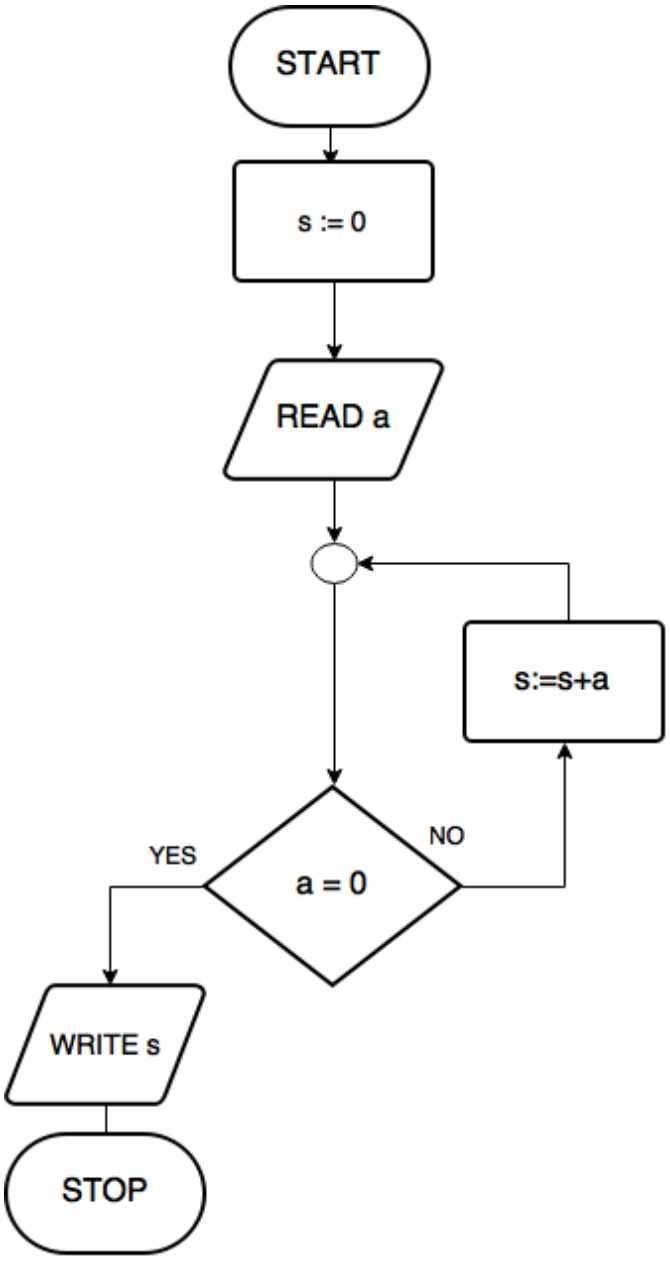


Linii de cod:

```
program determinant;{antet}

{blocul de declaratii si definitii:}
var
i,j:word;
  a:array[1..2,1..2] of real;
  d:real;

{corpul programului:}
begin
  for i:=1 to 2 do
  for j:=1 to 2 do
  begin
write('A',i,',',j,' = ');
  readln(a[i,j]);
end;

  d:=a[1,1]*a[2,2]-a[1,2]*a[2,1];

  writeln('Determinatul este D =
',d:8:4);

  readln;
end.
```

21. Sa se scrie un program care sa calculeze suma datelor introduse de la tastatură atât timp cât valoarea introdusă este diferită de 0. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

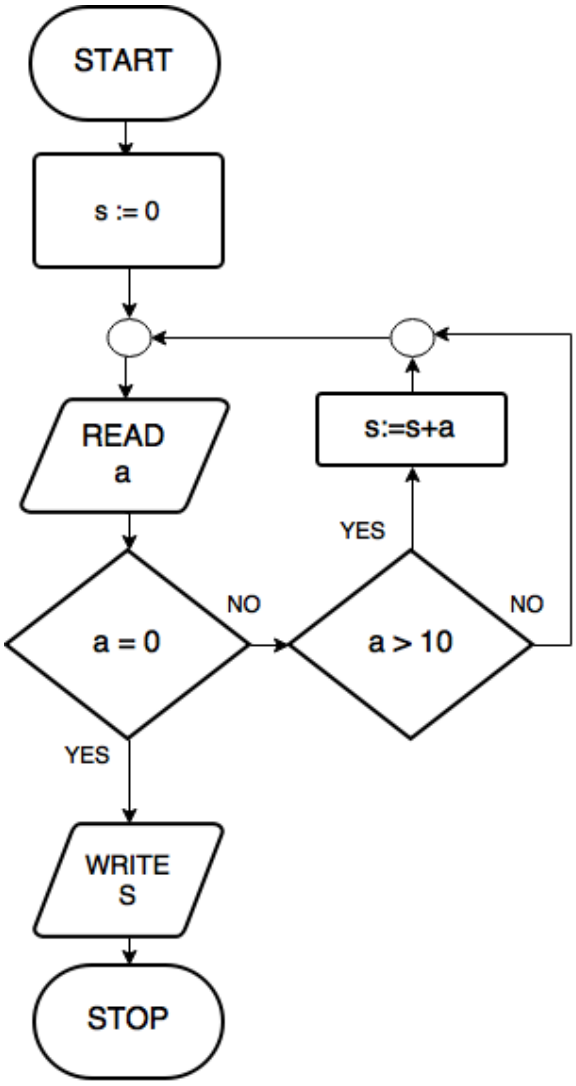


Linii de cod:

```
program
suma_dif_zero;{antet}

{blocul de declaratii si
definitii:}
var
i:word;{optional, indice
pentru afisare a, nu este
in schema logica}
  a,s:real;

{corpul programului:}
begin

s:=0; {initializare}
i:=1;

repeat
{intrucere a cu afisarea
unui indice optional:}
write('A',i,' = ');
readln(a);
  s:=s+a;
  i:=i+1
until a=0;

  writeln;
  writeln('Suma este S =
',s:8:4);

  readln;
end.
```

22. Sa se scrie un program care sa calculeze suma datelor introduse de la tastatură având valoarea mai mare decât 10. Cererea de date se va opri la introducerea unui numar negativ. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

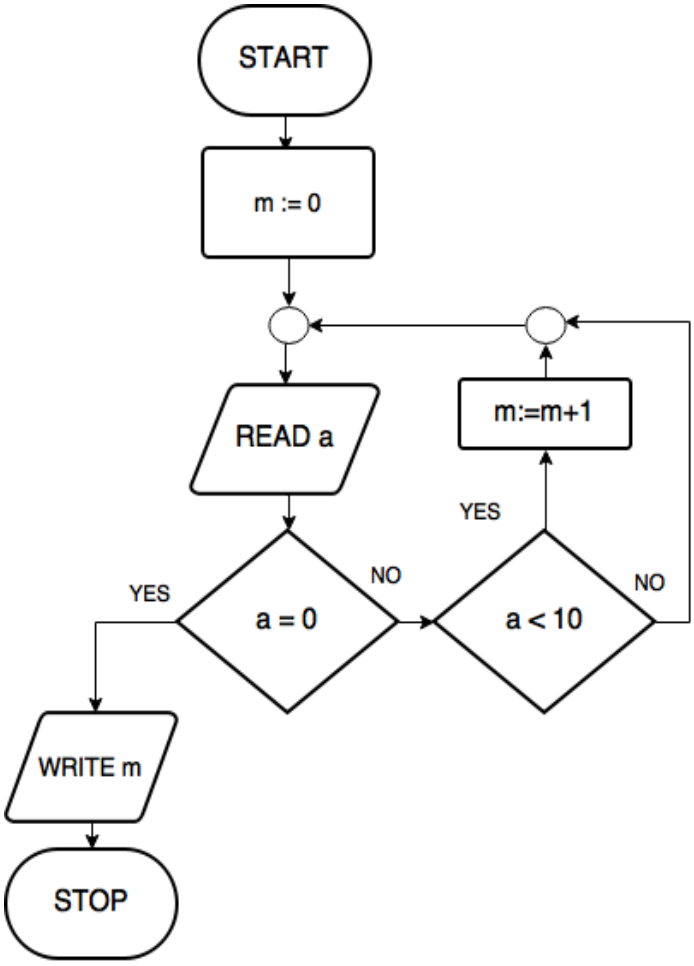


Linii de cod:

```
program
suma_peste_zece;{antet}

{blocul de declaratii si
definitii:}
var
i:word; {indice optional
pentru afisare a}
a,s:real;

{corpul programului:}
begin

s:=0; {initializare}
i:=1;

repeat
{intrucere a cu afisarea
unui indice optional:}
write('A',i,' = ');
readln(a);

if a>10 then
s:=s+a;

  i:=i+1
until a=0;

  writeln;
  writeln('Suma valorilor
> 10 este S = ',s:8:4);

  readln;
end.
```

23. Sa se scrie un program care sa calculeze numărul de datelor introduse de la tastatură având valoarea mai mică decât 10. Cererea de date se va opri la introducerea lui 0. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

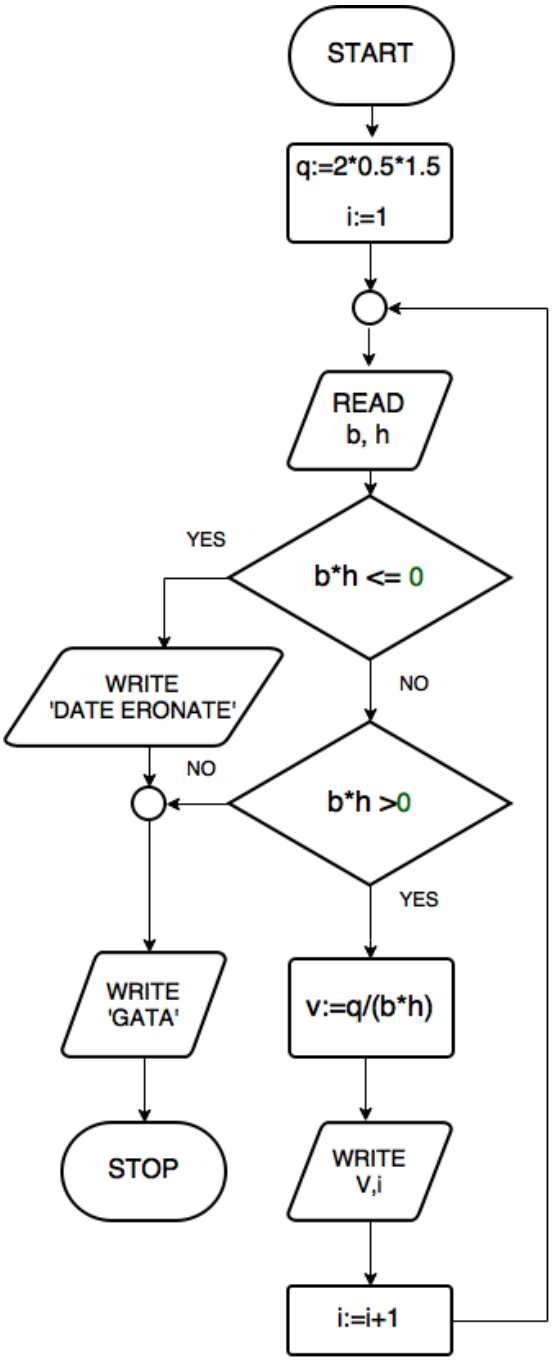


Linii de cod:

```
program
nr_mic_zece;{antet}

{blocul de declaratii si
definitii:}
var
i,m:byte; {indice
optional pentru afisarea
valorilor a, inexistent
in schema logica}
a:real;

{corpul programului:}
begin

m:=0; {initializare}
i:=1;

repeat
{intrucere a cu afisarea
unui indice optional:}
write('A',i,' = ');
readln(a);

if a<10 then
m:=m+1;

  i:=i+1
until a=0;

  writeln;
  writeln('Ati introdus
',m,' valori sub 10');

  readln;
end.
```

24. Ştiind că debitul ce se scurge prin doua secţiuni oarecare apartinând unei conducte este $Q=A_1*V_1= A_2*V_2$=constant, să se scrie un program care să calculeze viteza lichidului pentru orice secţiune aparţinând unei conducte de formă dreptunghiulară (mereu plină), cunoscând următoarele date pentru sectiunea de intrare (A_1) : b_1=2m, h_1=0.5m, V1=1.5m/s. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

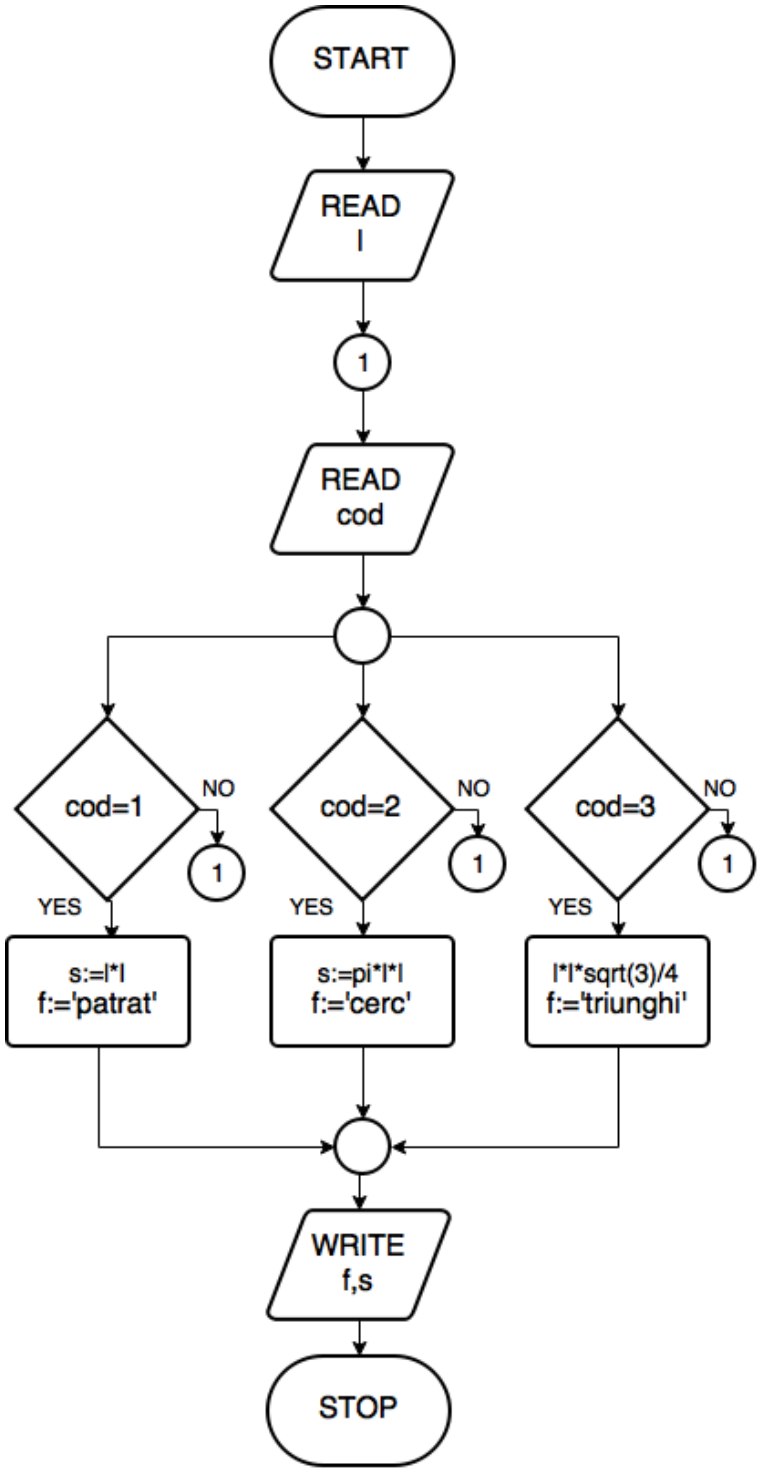


Linii de cod:

```
program Debit;{antet}

{blocul de declaratii si
definitii:}
var
i:word;
  q,b,h,v:real;

{corpul programului:}
begin

{initializare:}
q:=2*0.5*1.5;
i:=1;

repeat
{intrucere b si h cu
afisarea indicelui i:}
write('b',i,' = ');
readln(b);
  write('h',i,' = ');
readln(h);

if b*h >0 then
      begin
v:=q/(b*h);
      writeln('Viteza este
V = ',v:8:4);
      writeln;
      i:=i+1
end
    else writeln('valori
eronate !')

until b*h <= 0;
 writeln;
 writeln('GATA !')
end.
```

25. Să se scrie un program de calcul al ariei unei figuri geometrice: pătrat, cerc sau triunghi echilateral. Mărimea x este interpretată ca latura pătratului, raza cercului sau latura triunghiului (se vor introduce coduri pentru figuri, de exemplu: pătrat – 1, cerc – 2, triunghi – 3). Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:



Linii de cod:

```
program fig_geom;
var
  l,s:real;
  cod: byte;
  f:string;

  begin
  write('Introduceti latura/raza = '); readln(l);writeln;
  writeln('cod 1 = patrat'); writeln('cod 2 = cerc' );
  writeln('cod 3 = triunghi' );

  {alegere cod dintre valorile cerute:}
  repeat
       repeat
       write('alegeti cod : ');readln(cod);
       until cod<4
  until cod>0;
  writeln;

  {calcul arie si atribuire nume figura pt var f:}
  case cod of
  1: begin s:=l*l; f:='patrat' end;
  2: begin s:=pi*l*l; f:='cerc' end;
  3: begin s:=l*l*sqrt(3)/4 ; f:='triunghi' end
  end;

  {afisare rezultat:}
  writeln('Pentru ',f,' aria = ', s:6:3);
  readln

  end.
```

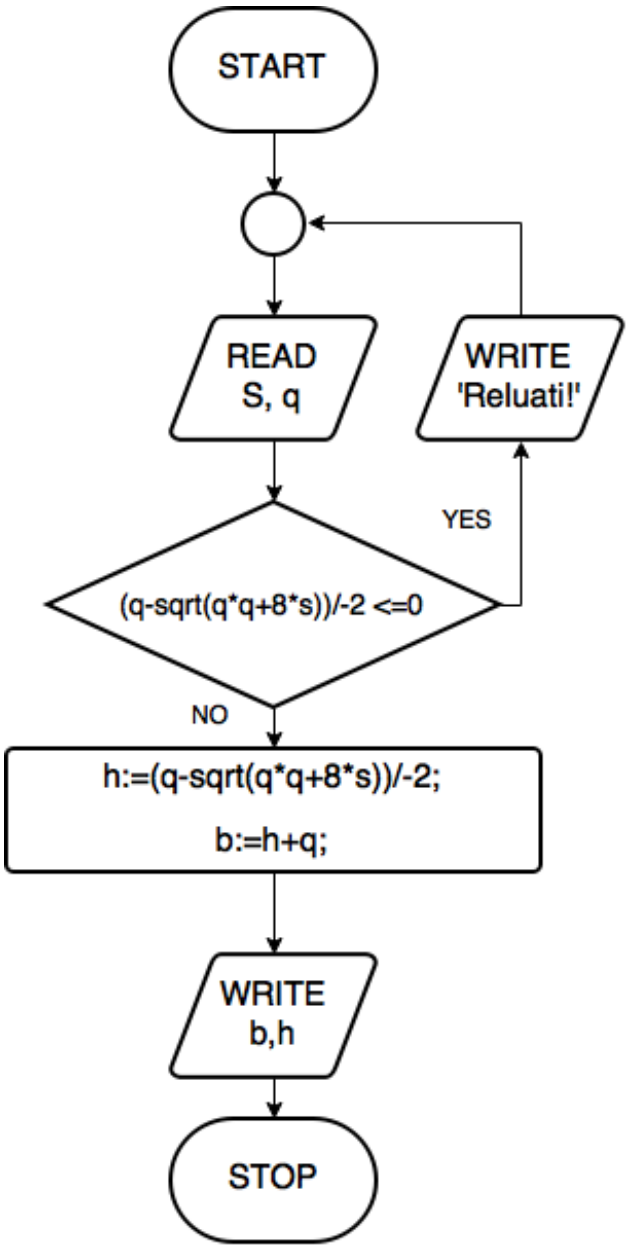
26. Să se scrie un program de determinare a înălţimii (H) unui triunghi ştiindu-se aria (A) şi lungimea bazei mai mare decât înălţimea cu q unităţi. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Rezolvare:

Baza triunghiului =B
B=H+q
A=B*H/2
A=(H+q)*H/2
$-H^2$-q*H+2*A=0 , ecuatie grad 2, cu necunoscuta H avand valoarea:

$$H=\frac{-(-q)-\sqrt{(-q)^2-4*(-1)*2*A}}{2*(-1)}$$

Schema logică:

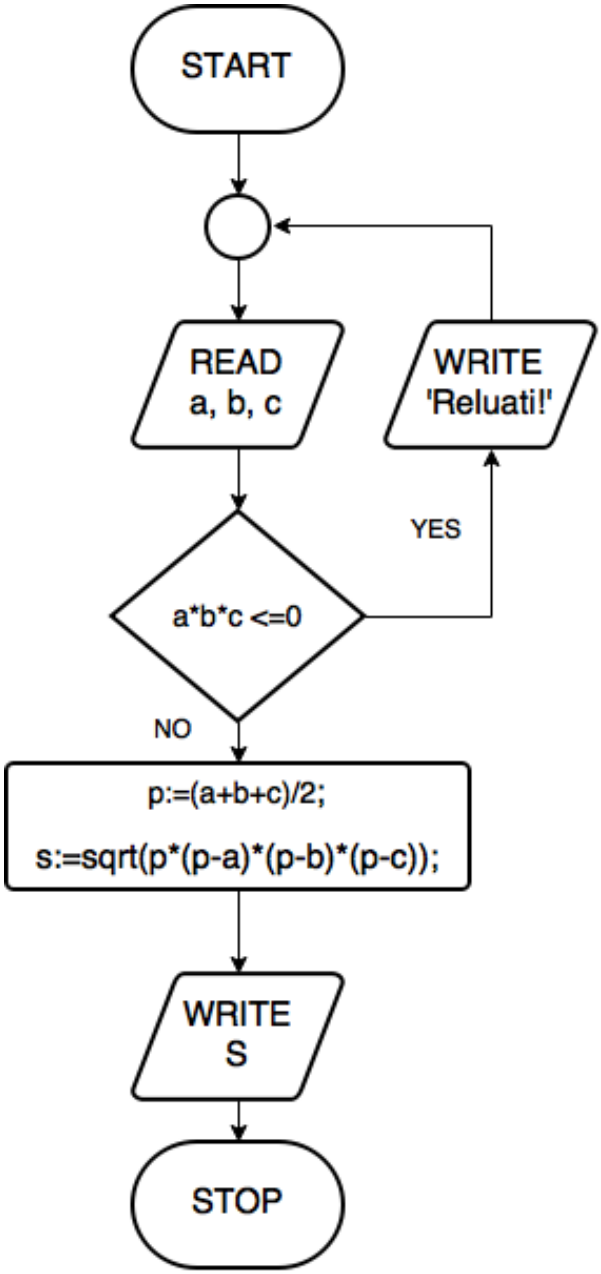


Linii de cod:

```
program triunghi;
var
  q,s,b,h:real;

  begin

  {Introducere date cu
validare:}
  repeat
  write('Introduceti aria
S = '); readln(s);
  write('Introduceti
diferenta B-H= ');
readln(q);

    {mesaj eroare:}
    if (q-sqrt(q*q+8*s))/-
2 <=0 then
writeln('Reluati, valori
eronate !')

  until (q-
sqrt(q*q+8*s))/-2 >0;

  {calcul inaltime si baza
triunghi:}
  h:=(q-sqrt(q*q+8*s))/-2;
  b:=h+q;

  {afisare rezultate:}
  writeln('H = ', h);
  writeln('B = ', b);
  readln

  end.
```

27. Să se scrie un program de calcul al ariei unui triunghi după metoda lui Heron (A = $\sqrt{p(p-a)(p-b)(p-c)}$, unde a, b, c sunt laturile triunghiului, iar p semiperimetrul său). Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

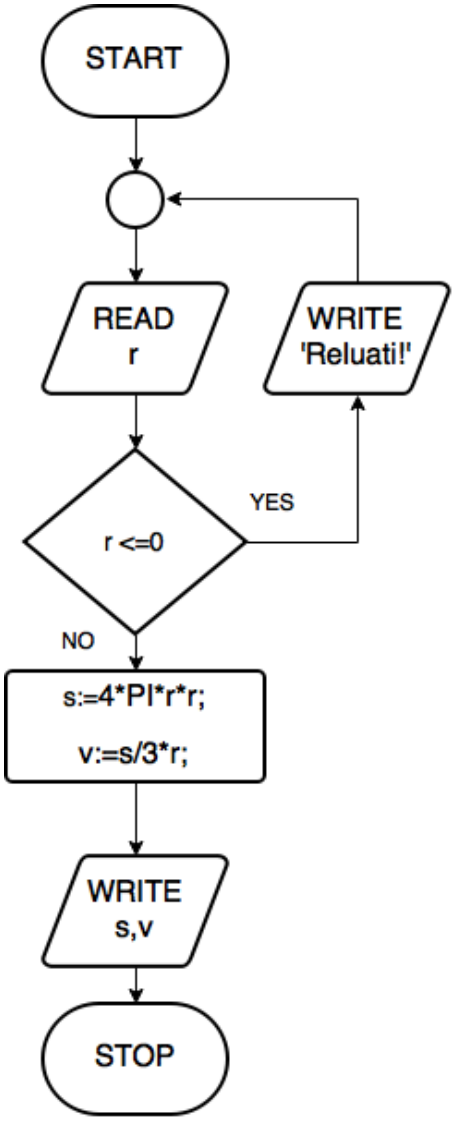


Linii de cod:

```
program Heron;
var
  p,a,b,c,s:real;

  begin

  {Introducere date, se
refuza valorile <= 0:}
  repeat
  write('Introduceti a = ');
readln(a);
  write('Introduceti b = ');
readln(b);
  write('Introduceti c = ');
readln(c);

    {mesaj eroare:}
    if a*b*c <=0 then
writeln('Reluati, valori
eronate !');
  writeln

  until a*b*c >0;

  {calcul semiperimetru si
arie triunghi:}
  p:=(a+b+c)/2;
  s:=sqrt(p*(p-a)*(p-b)*(p-
c));

  {afisare rezultate:}
  writeln('Aria = ', s:6:3);
  readln

  end.
```

28. Să se scrie un program care să calculeze aria şi volumul unei sfere de rază R (număr întreg). Toate cererile şi rezultatele vor fi afişate explicit pe ecran. (A_{sf}= 4*π*R^2, V_{sf} = 4*π*R^3/3)

Schema logică:



Linii de cod:

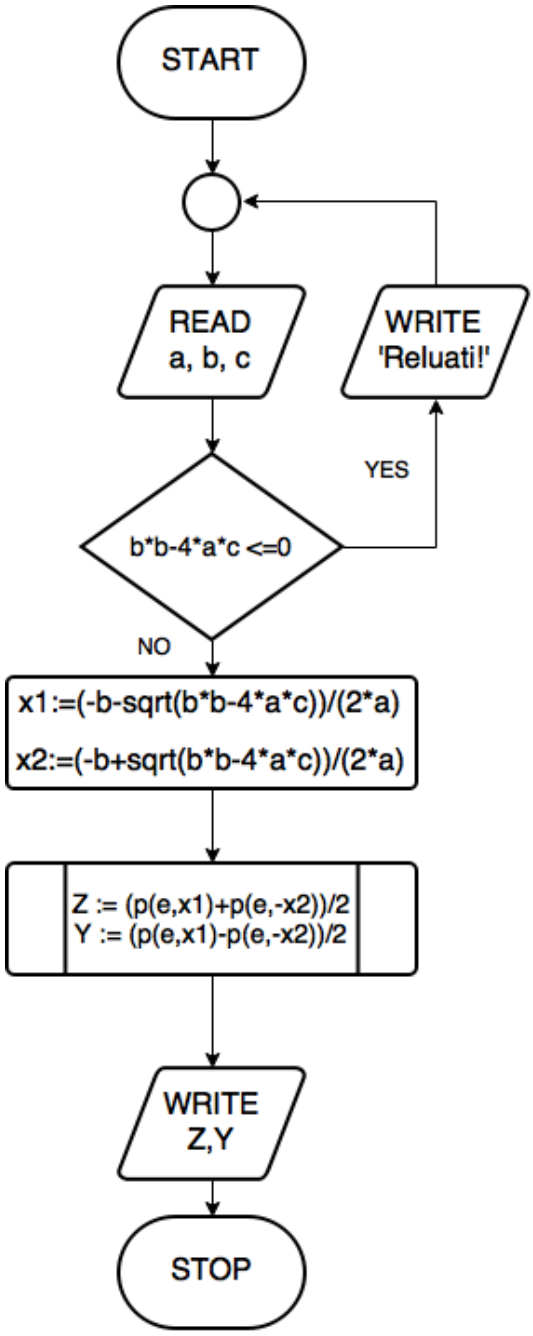
```
program sfera;{antet}

{blocul de declaratii si
definitii:}
var
r,s,v:real;

{corpul programului:}
begin


repeat
{intrucere r:}
write('r = '); readln(r);

if r<=0 then writeln('Valoare
introdusa este eronata,
reluati !')
until r>0;

s:=4*PI*r*r;
v:=s*r/3;

  writeln;
  writeln('Aria este S =
',s:8:4);
  writeln('Volumul este v =
',v:8:4);
  readln;

end.
```

29. Să se scrie un program de calcul al valorilor funcţiilor: Z = ($e^{x_1} + e^{-x_2}$)/2 , Y = ($e^{x_1} - e^{-x_2}$)/2, dacă x_1, x_2 reprezintă rădăcinile ecuaţiei de gradul 2 (scrisă în forma generală, cu coeficienţii *a*, *b*, *c* introduşi de la tastatră). Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Rezolvare:
Pentru a avea doua rădăcini distincte punem condiția, la introducerea datelor in program, ca DELTA > 0, adică b^2-4ac > 0.
Rădăcinile x_1, x_2 sunt date de relația cunoscuta:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Schema logică:



Linii de cod:

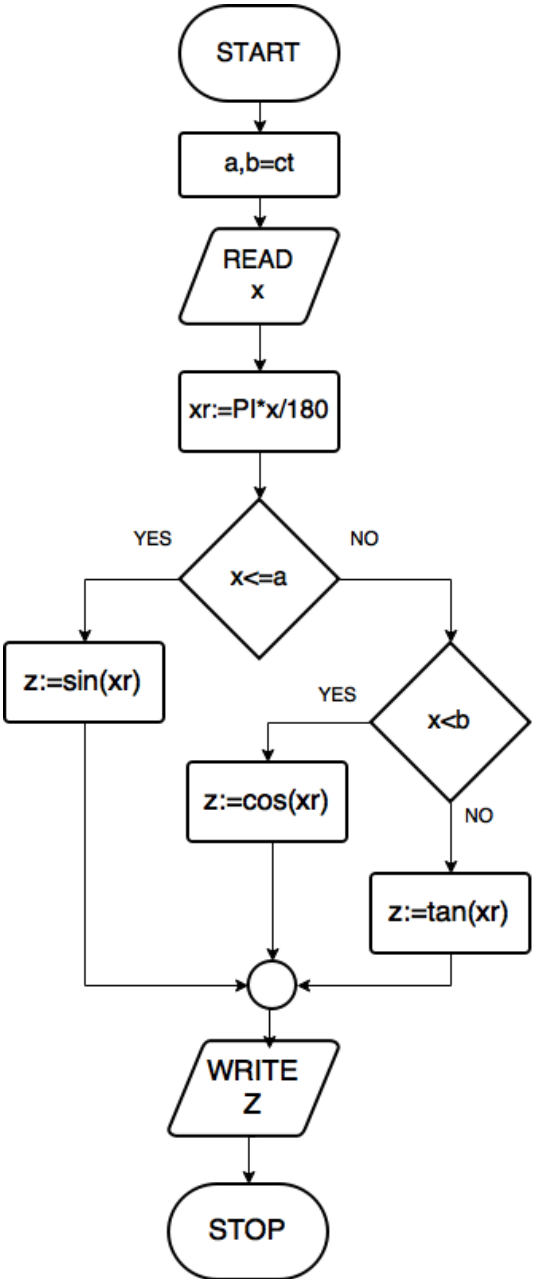
```
program functii;{antet}

{la variantele mai vechi de
Pascal, e trebuie definit ca si
constanta}

{blocul de declaratii si
definitii:}
var a,b,c,z,y,x1,x2:real;

{definire functie pentru calculul
puterii:}
function p(u,v:real):real;
begin
p:=exp(v*ln(u));
end;

begin
repeat
write('a =');readln(a);
write('b =');readln(b);
write('c =');readln(c);
if b*b-4*a*c<=0 then
writeln('Eroare, alte valori !');
writeln
until b*b-4*a*c>0;

x1:=(-b-sqrt(b*b-4*a*c))/(2*a);
x2:=(-b+sqrt(b*b-4*a*c))/(2*a);
z:=(p(e,x1)+p(e,-x2))/2;
y:=(p(e,x1)-p(e,-x2))/2;

{afisare rezultate:}
writeln('Z = ',z);
writeln('Y = ',y);
end.
```

30. Să se scrie un program care calculează valoarea funcției z pentru diferite valori ale argumentului x:

$$z = \begin{cases} \sin x & ,dacă\ x \le a \\ \cos x & ,dacă\ a < x < b \\ tgx & ,dacă\ x \ge b \end{cases}$$, unde a, b – constante fixate

Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

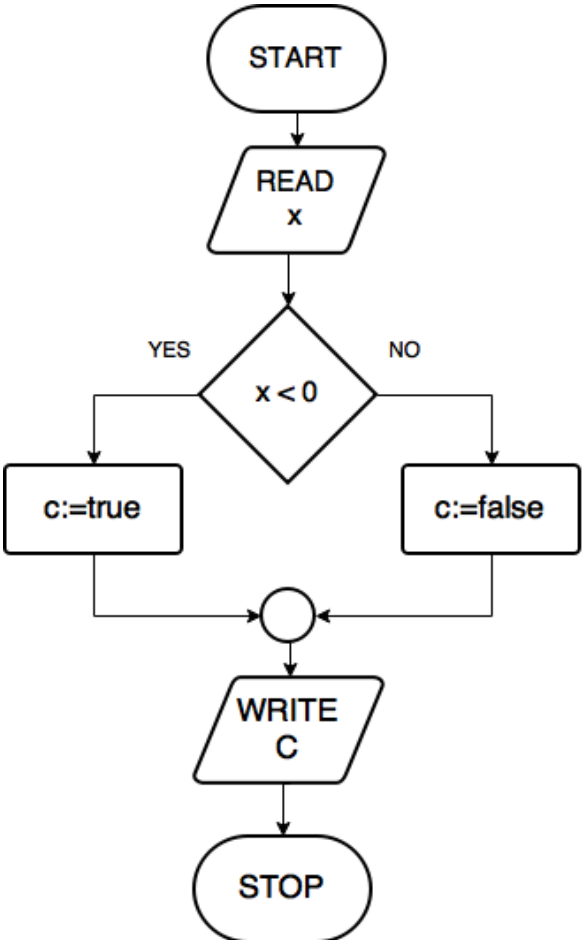


Linii de cod:

```
program functie;

{declaratie constante impuse:}
const
  a=30.00;
  b=60.00;

var
  x,xr,z:real;
  f:string;{pentru afisarea
sugestiva a rezultatelor}

begin
{introducere date:}
write('Introduceti valoarea in
grade, X = ');
readln(x);

{calcul:}
xr:=PI*x/180;
if x<=a then begin z:=sin(xr);
f:='sin x = ' end else
if x<b  then begin z:=cos(xr);
f:='cos x = ' end else
begin z:=sin(xr)/cos(xr);
f:='tg x = ' end;

writeln(f,z:6:3)

end.
```

31. Să se scrie un program de calcul al valorilor funcţiei: C = $\begin{cases} true, & dacă\ x < 0 \\ false, & dacă\ x \geq 0 \end{cases}$ Toate cererile şi rezultatele vor fi afişate explicit pe ecran

Schema logică:



Linii de cod:

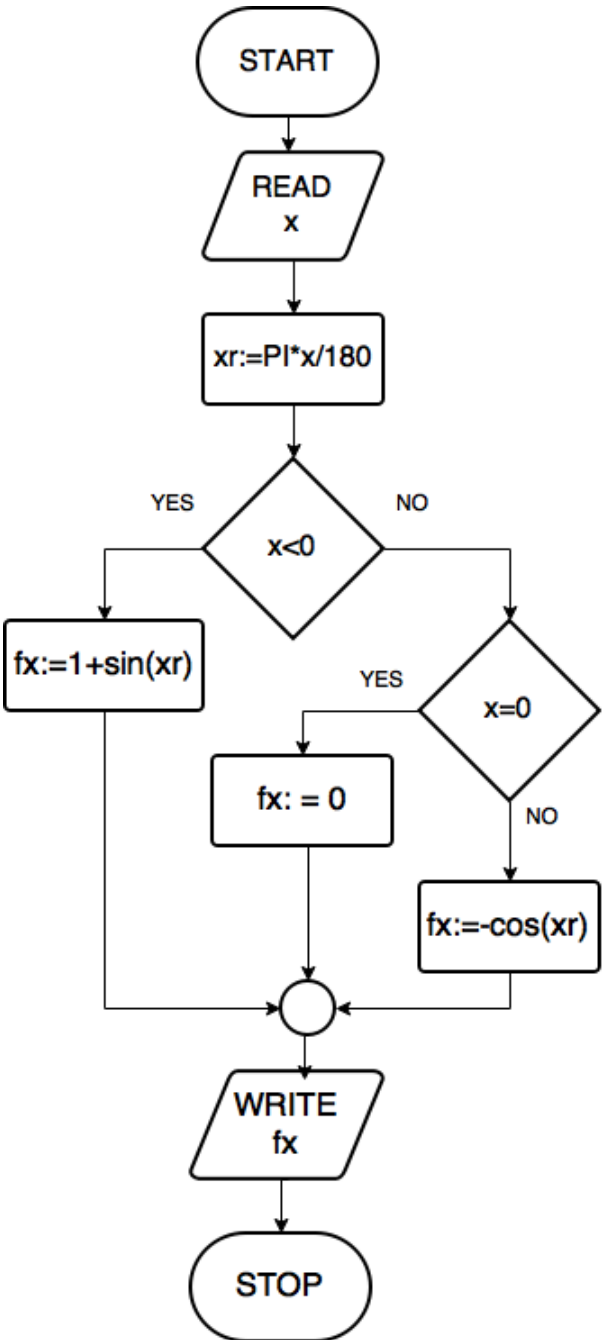
```
program logic;

var
  c:boolean;
  x:real;

begin
{introducere date:}
write('Introduceti valoarea
X = ');
readln(x);

{bloc de calcul:}
if x<0 then c:=true else
c:=false;

{afisare rezultat:}
writeln('C = ',c)

end.
```

32. Dată fiind valoarea argumentului *x*, să se scrie un program de calcul al funcţiei:

f(x) = $\begin{cases} 1+\sin x, & dacă\ x < 0 \\ 0, & dacă\ x = 0 \\ -\cos x, & dacă\ x > 0 \end{cases}$

Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

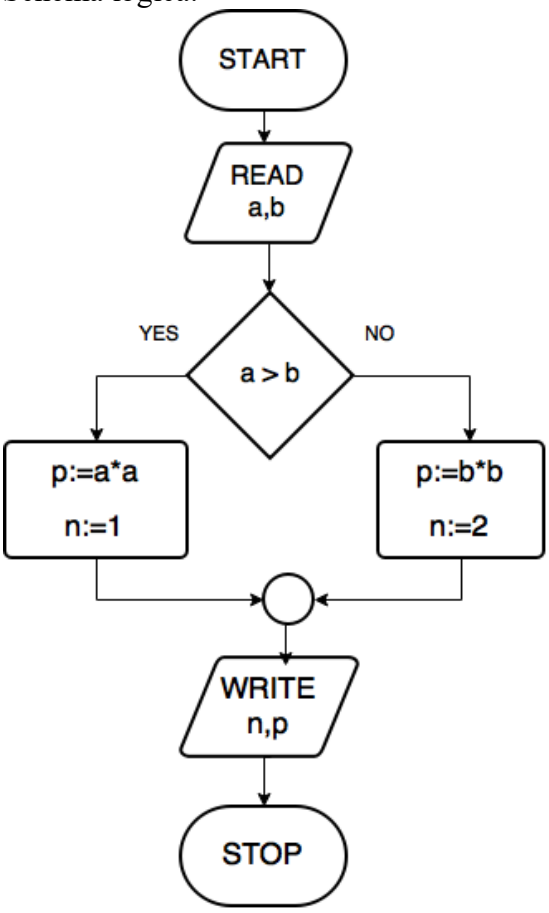


Linii de cod:

```
program functie;

{declarati variabile:}
var
  x,xr,fx:real;

begin
{introducere date:}
write('Introduceti X = ');
readln(x);

xr:=PI*x/180; {radiani}

{calcul:}
if x<0 then fx:=1+sin(xr) else
if x=0 then fx:=0
else
fx:=-cos(xr);

writeln('F(x) = ',fx:6:3)

end.
```

33. Să se scrie un program care calculează pătratul celui mai mare din numerele *a* şi *b* şi impuneţi identificatorul N=1 dacă mai mare este *a* şi N=2 în caz contrar. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

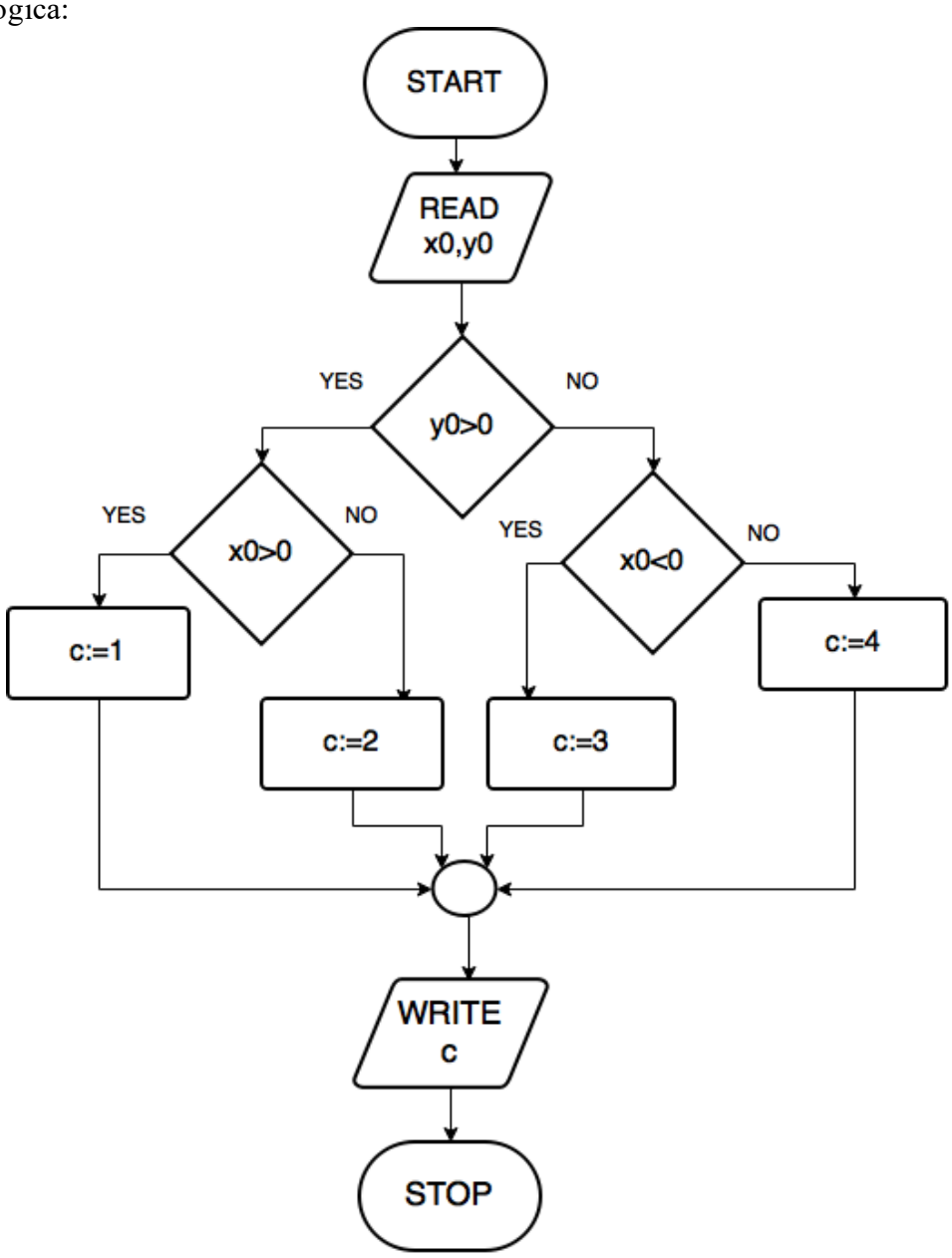


Linii de cod:

```
program functie_max_ab;

{declarati variabile:}
var
  a,b,p:real;
  n:byte;

begin
{introducere date:}
write('Introduceti a = ');
readln(a);
write('Introduceti b = ');
readln(b);

{calcul:}
if a>b then begin p:=a*a; n:=1
end else begin p:=b*b; n:=2
end;

writeln('Pentru valoarea mai
mare, indicativ N=',n,',
patratul este = ',p:6:3)

end.
```

34. Să se scrie un program ce determină cărui cadran îi aparţine punctul de coordonate (x_0, y_0) şi afişează numărul cadranului. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logica:



Linii de cod:

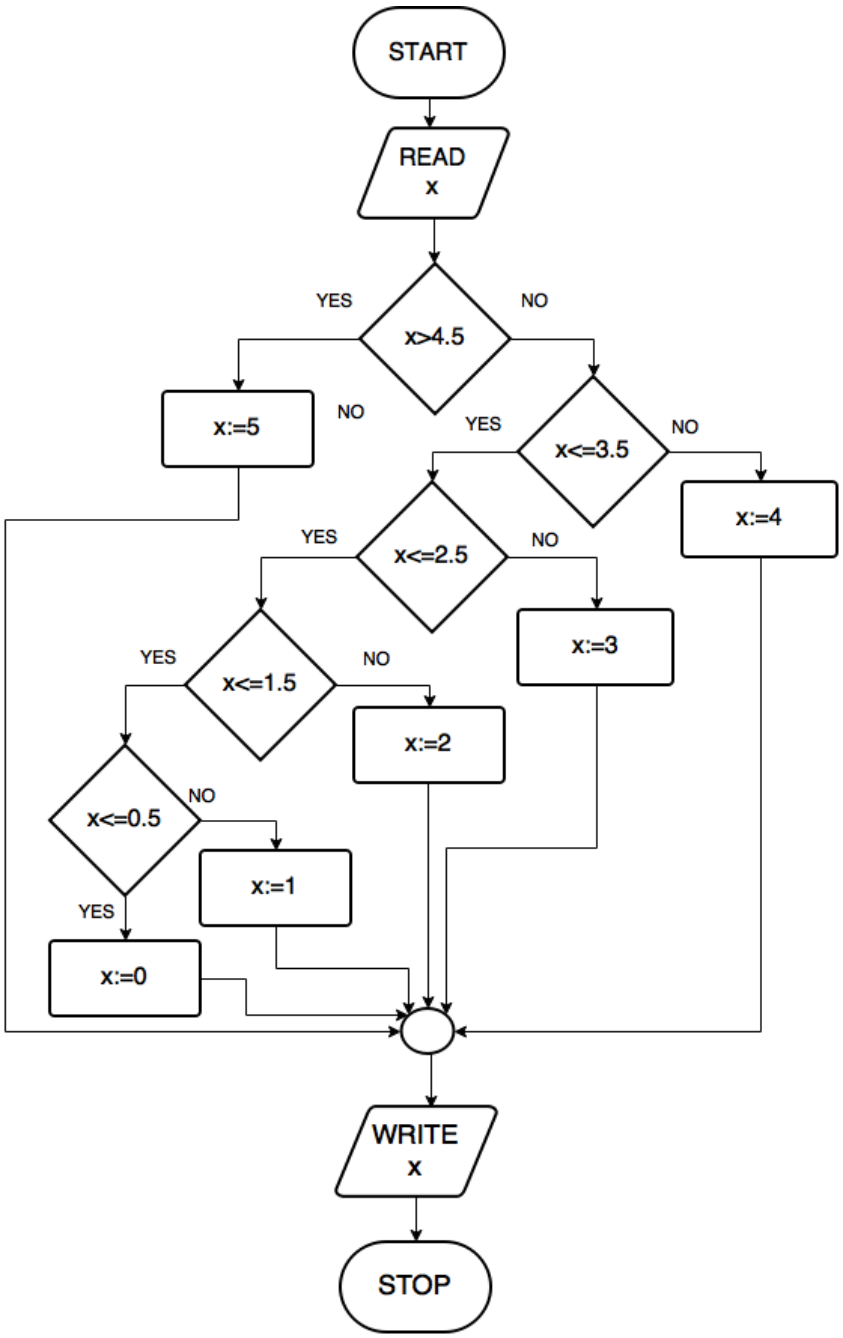
```
program cadran;
{declaratii variabile:}
var
  x0,y0:real;
  c:byte;
begin
write('Introduceti Xo = ');
readln(x0);
write('Introduceti Yo = ');
readln(y0);
{bloc de calcul:}
if y0>0 then
  if x0>0 then c:=1
  else c:=2
else
  if x0<0 then c:=3
  else c:=4;
writeln('Cadranul ',c);
readln;

end.
```

35. Rotunjiţi numărul pozitiv x mai mic decât 5 până la cel mai apropiat număr întreg:

0, dacă x ≤0.5; 3, dacă 2.5<x≤3.5
1, dacă 0.5<x≤1.5 4, dacă 3.5<x≤4.5
2, dacă 1.5<x≤2.5 5, dacă 4.5<x.

Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

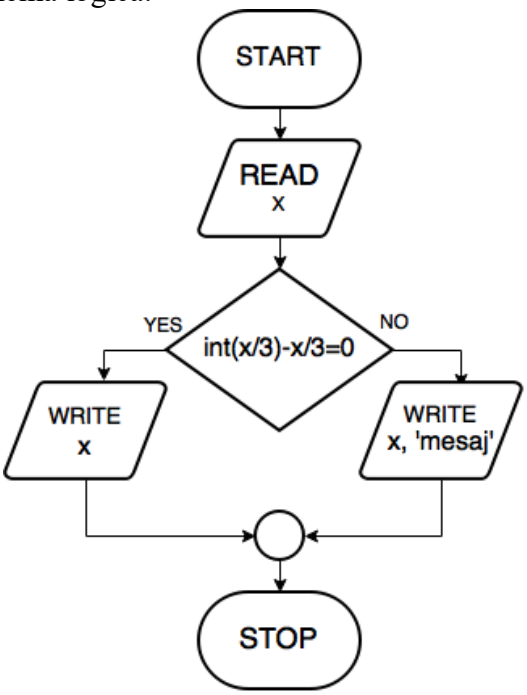
Schema logica:



```
Program rot_interv;
var x:real;

begin
write('Introduceti X = ');
readln(x);

if x>4.5 then x:=5
else
  if x<=3.5 then
    if x<=2.5 then
      if x<=1.5 then
        if x<=0.5 then
        x:=0
        else
        x:=1
      else
      x:=2
    else
    x:=3
  else
  x:=4;

writeln('X = ',x);
readln

end.
```

36. Să se scrie un program ce determină dacă valoarea variabilei de tip întreg x are divizor pe 3. Dacă această relaţie are loc, se va afişa valoarea lui x, altfel se va afişa mesajul „[valoarea lui x] – nu are printre divizori cifra 3”. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:



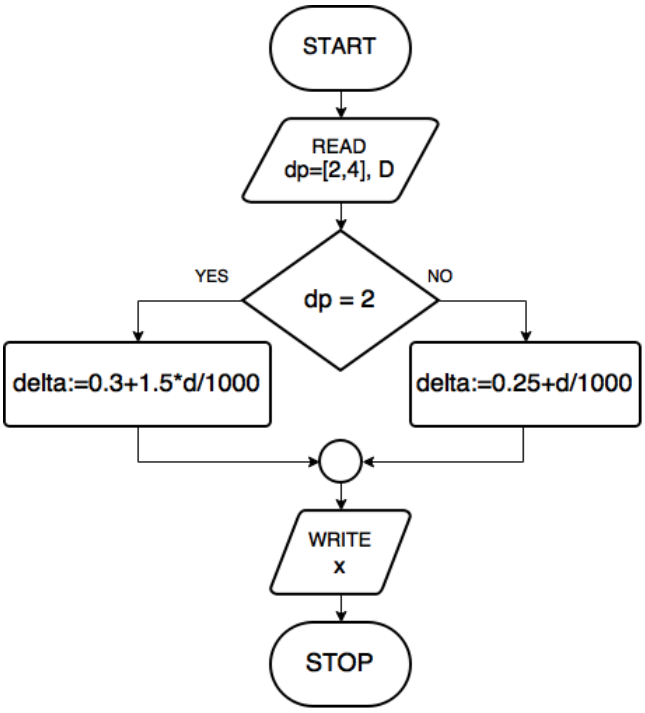
Linii de cod:

```
program div_3;

{Bloc declaratii:}
var x:integer;

begin
write('Introduceti X =
');readln(x);

if int(x/3)-x/3=0 then
writeln('Valoarea X = ',x)
else
writeln(x,' nu are printre
divizori cifra 3');

readln
end.
```

37. Mărimea întrefierului δ al motorului asincron poate fi determinat după formula empirică:

$$\delta = \begin{cases} 0.3+1.5D/1000 & ,dacă\ 2p=2 \\ 0.25+D/1000 & ,dacă\ 2p=4 \end{cases}$$, unde D diametrul interior al miezului statorului. Să se scrie un program de calcul al mărimii întrefierului pentru diferite mărimi D şi perechi de poli p. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:



Linii de cod:

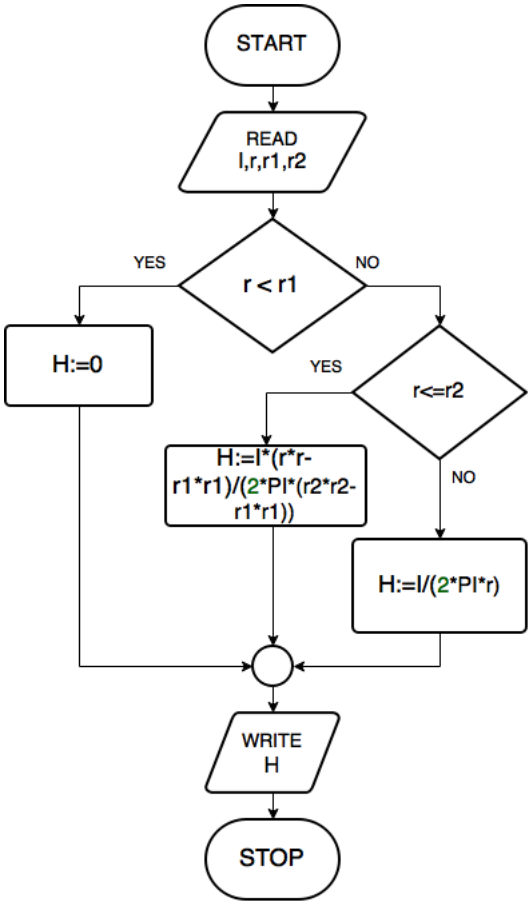
```
program intrefier;

var dp: byte;
    d, delta: real;

begin
{introducere date:}
write('Introdu diametru stator
D = ' );
readln(d);
{bloc introducere dp cu
verificare domeniu 2 sau 4:}
repeat
write('Introdu nr. perechi de
poli 2p = ' );
readln(dp);
    if dp<>2 then
    if dp<>4 then writeln
    ('Eroare, reluam!');
until (dp=2) or (dp=4);

{bloc calcul:}
if dp=2 then
delta:=0.3+1.5*d/1000
else delta:=0.25+d/1000;

{bloc afisare rezultat:}
writeln('Pentru ',dp,' poli,
si D = ',d,' intrefierul este
= ',delta);
readln
end.
```

38. Să se scrie un program de calcul al intensităţii câmpului magnetic cauzat de circulaţia curentului continuu I într-o ţeavă cu raza internă r_1 şi raza externă r_2:

$$H = \begin{cases} 0 & , dacă\ r < r_1 \\ \dfrac{I(r^2 - r_1^2)}{2\pi r(r_2^2 - r_1^2)} & , dacă\ r_1 \le r \le r_2 \\ I / 2\pi r & , dacă\ r > r_2 \end{cases}$$

Toate cererile şi rezultatele vor fi afişate explicit pe ecran..

Schema logică:

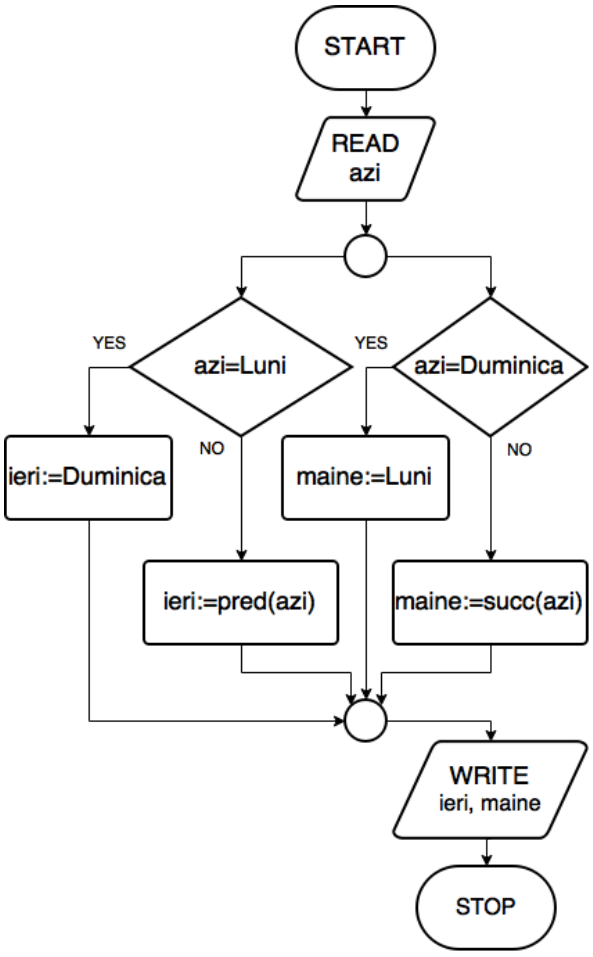


Linii de cod:

```
program camp_magnetic;
var I,r,r1,r2,h: real;

begin
{introducere date:}
write('Introdu intensitatea
curentului I = ' );
readln(I);
write('Introdu raza interioara
R1 = ' );
readln(r1);

repeat
write('Introdu raza exterioara
R2 = ' );
readln(r2);
if r2<=r1 then
writeln('Eroare, reluati !');
until r2>r1;

write('Introdu raza curenta R
= ' );readln(r);

{bloc calcul:}
if r<r1 then
h:=0
else
  if r<=r2 then
  h:=I*(r*r-
r1*r1)/(2*PI*(r2*r2-r1*r1))
  else
  h:=I/(2*PI*r);
{bloc afisare rezultat:}
writeln('Intensitatea campului
H = ',H);
readln
end.
```

39. Să se scrie un program care calculează valorile variabilelor *ieri* şi *mâine*, cunoscându-se *azi*. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

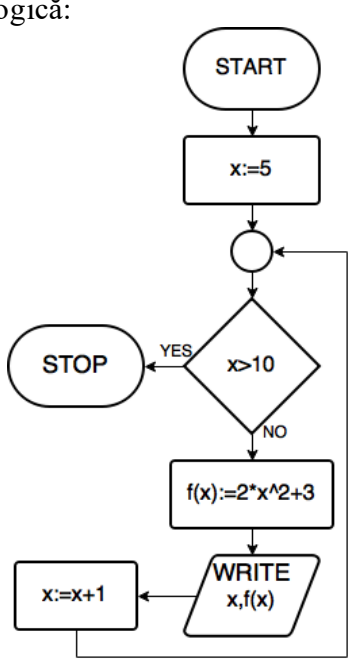


Linii de cod:

```
program azi_maine;
type zile=(Luni, Marti,
Miercuri, Joi, Vineri, Sambata,
Duminica);
var ieri, maine, i:byte;

begin
{Bloc introducere date:}
writeln('Alegeti pentru "AZI"
numarul corespunzator zilei din
saptamana:');
for i:=0 to 6 do
writeln(zile(i):8,' = ',i+1 );
writeln;
repeat
write('Numar ales [in intervalul
1 - 7] : '); readln(i);
until (i>0) and (i<8);

{Bloc calcul:}
i:=i-1;
if i=0 then ieri:=6 {Duminica}
else
ieri:=pred(i);

if i=6 then maine:=0 {Luni}
else
maine:=succ(i);

{Bloc afisare rezultat:}
writeln('Azi = ':8,zile(i));
writeln('Maine = ':8,
zile(maine));
writeln('Ieri = ':8,zile(ieri));
readln
end.
```

40. Să se scrie un program pentru tabelarea funcţiei f: N→N, f = $2 \cdot x^2+3$ pentru valori ale lui x cuprinse în intervalul [5,10]. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

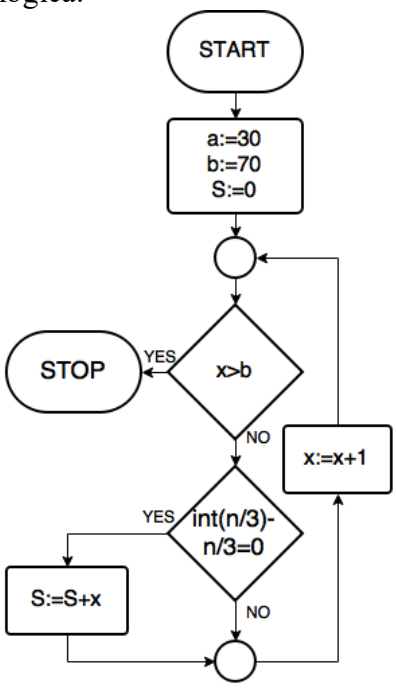


Linii de cod:

```
program tab_func;
var f,x:byte;
begin
writeln('tabel functie
x=(5..10):' );
for x:=5 to 10 do
writeln('x=',x:2,'
f(x)=',2*x*x+3);{sau rezervat
doua spatii pt x pentru
aliniere unitati-zeci}
writeln('gata !')
end.
```

41. Să se scrie un program de calul al sumei tuturor numerelor divizibile cu 3 din intervalul [30, 70]. Generalizare pentru intervalul [a, b], a, b – fixate. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

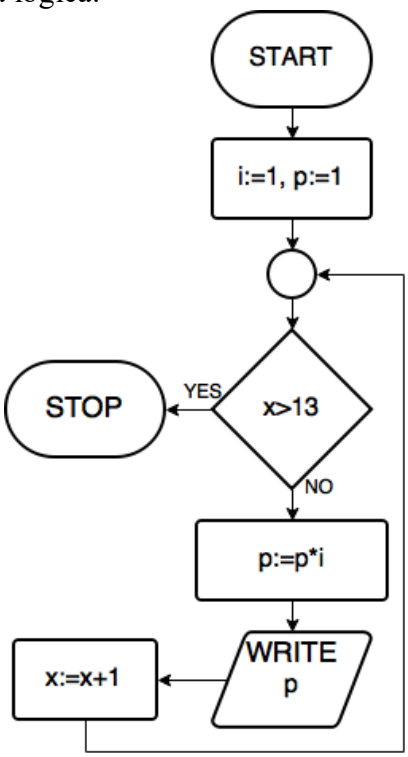


Linii de cod:

```
program sum_div3;
const a=30;
      b=70;
var n,s:byte;

begin
s:=0;
for n:=a to b do
if int(n/3)-n/3=0 then
s:=s+n;

writeln('Suma numerelor
divizibile prin trei in
intervalul [',a,'..',b,'] este
s=',s);
writeln('gata !')

end.
```

42. Să se scrie un program de determinare a produsului numerelor întregi de la 1 până la 13. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

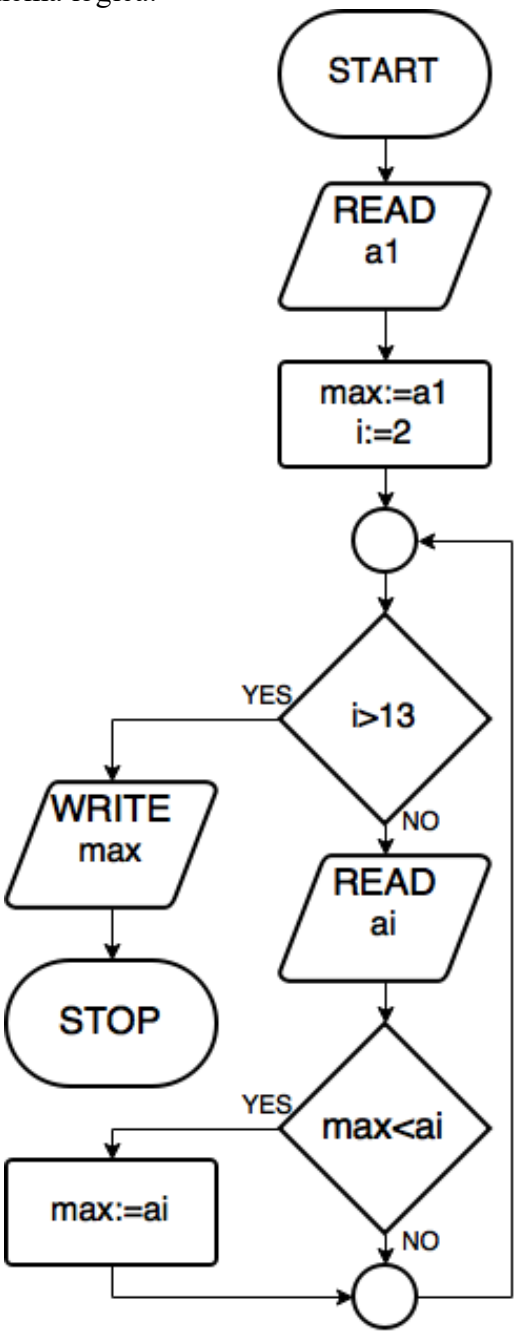


Linii de cod:

```
program prod13;

var n,p:word;

begin

{initialiare:}
p:=1;

{calcul:}
for n:=1 to 13 do
p:=p*n;

{afisare:}
writeln('Produsul numerelor
intregi pana la 13 este
p=',p);
writeln('gata !')

end.
```

43. Să se scrie un program care să calculeze maximul a 13 numere întregi. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

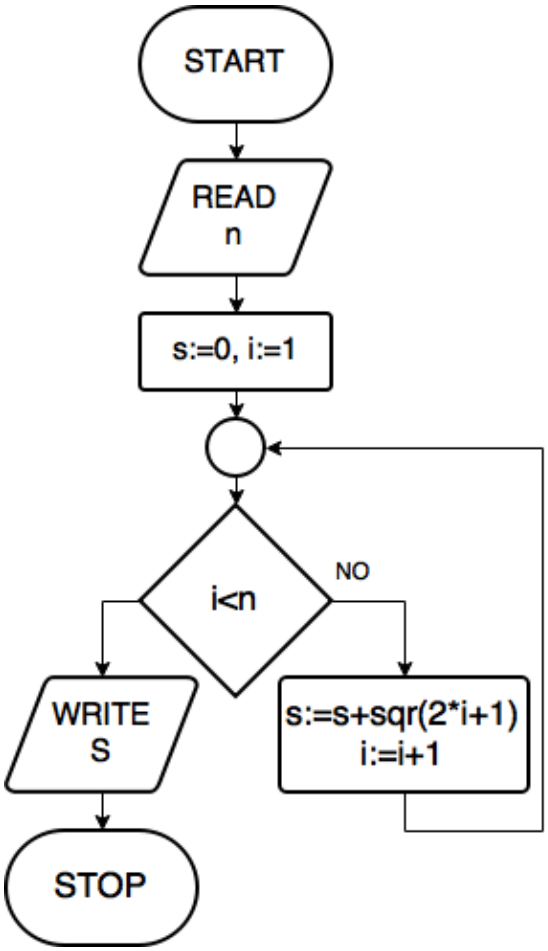


Linii de cod:

```
program max13;

var n,a, max:word;

begin

{initialiare:}
write('n1=');readln(max);{pres
upunem ca este si primul numar
introdus}

{introducere date si calcul:}
for n:=2 to 13 do
begin
write('n',n,'=');readln(a);
if a>max then max:=a;
end;

{afisare rezultat:}
writeln('Maximul este max
=',max);
readln
end.
```

44. Să se scrie un program de calcul al sumei $S = 1^2 + 3^2 + ... + (2n+1)^2$. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

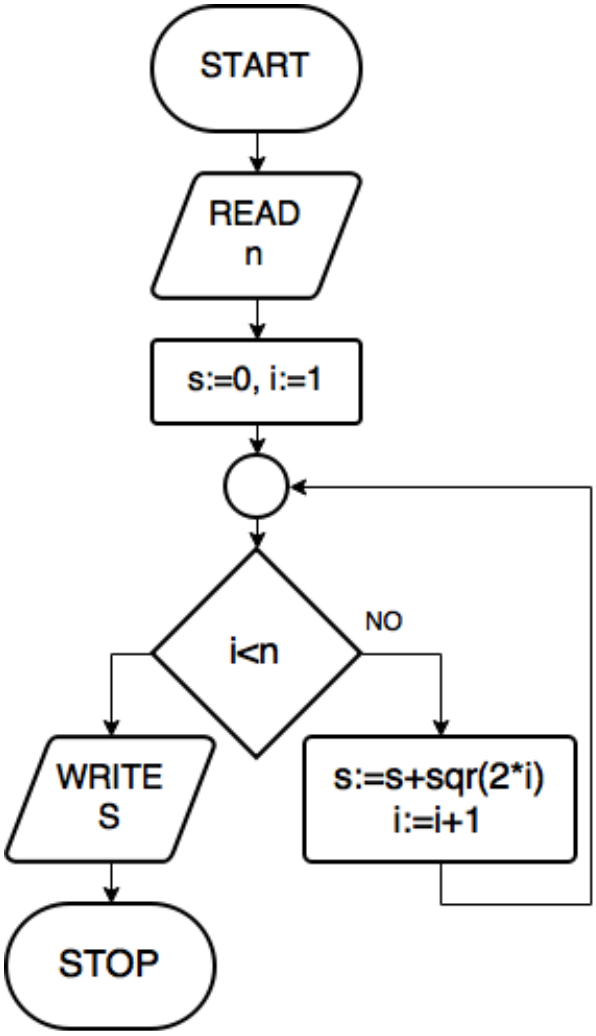


Linii de cod:

```
program sumaimp;

var n,i,s:word;

begin
{Introducere date:}
write('Introduceti N =
');readln(n);

{initialiare:}
s:=0;

{calcul:}
for i:=1 to n do
s:=s+sqr(2*i+1);

{afisare rezultat:}
writeln('Suma este s =',s);
readln
end.
```

45. Să se scrie un program de calcul al sumei $S = 2^2 + 4^2 + ... + (2n)^2$. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

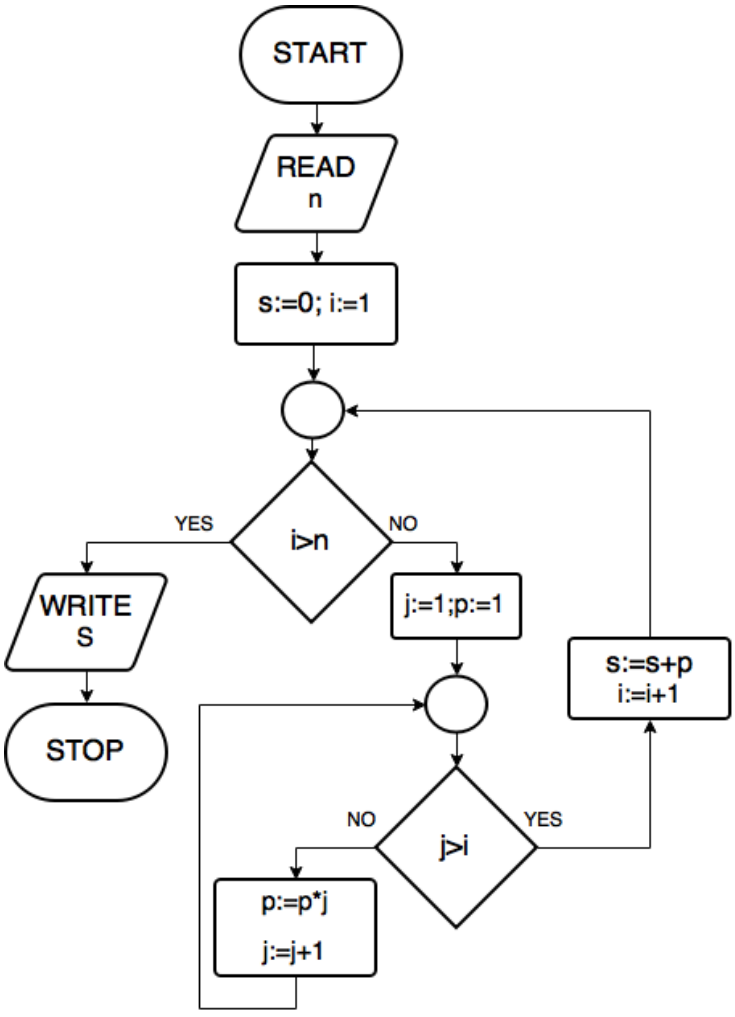


Linii de cod:

```
program sumapar;

var n,i,s:word;

begin
{Introducere date:}
write('Introduceti N =
');readln(n);

{initialiare:}
s:=0;

{calcul:}
for i:=1 to n do
s:=s+sqr(2*i);

{afisare rezultat:}
writeln('Suma este s =',s);
readln
end.
```

46. Să se scrie un program de calcul al sumei S = 1+ 1·2 + 1·2·3 + ... + 1·2·3·....·n Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

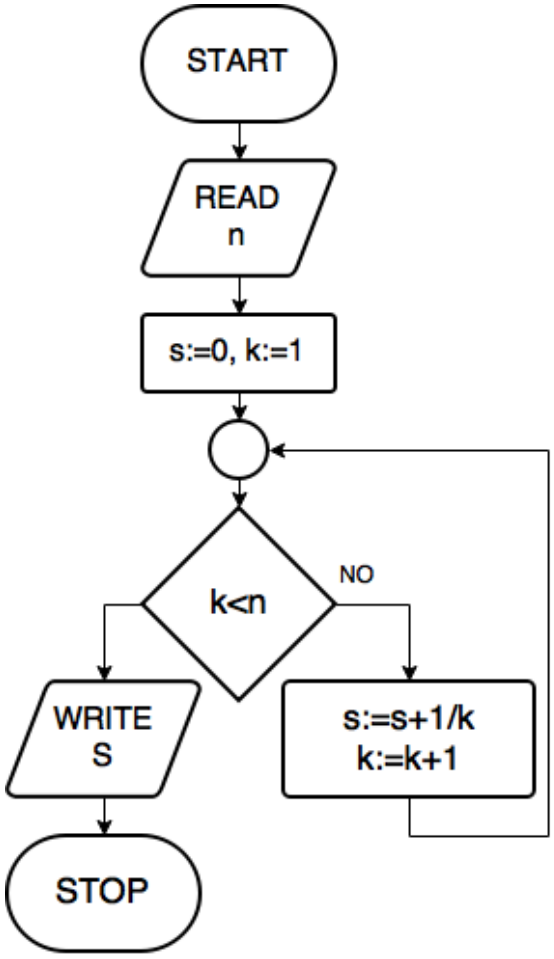
Schema logică:




```
program sumafac;

var n,i,j,s,p:word;

begin
{Introducere date:}
write('Introduceti N =
');readln(n);

{initialiare:}
s:=0;
p:=1;

{calcul:}
for i:=1 to n do
begin
p:=1;
for j:=1 to i do
p:=p*j;
s:=s+p
end;

{afisare rezultat:}
writeln('Suma este s
=',s);
readln
end.
```

47. Să se scrie un program pentru calulul sumei seriei finite S = $\sum_{k=1}^{N} \frac{1}{k}$, cu N un număr natural introdus de la tastatură. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

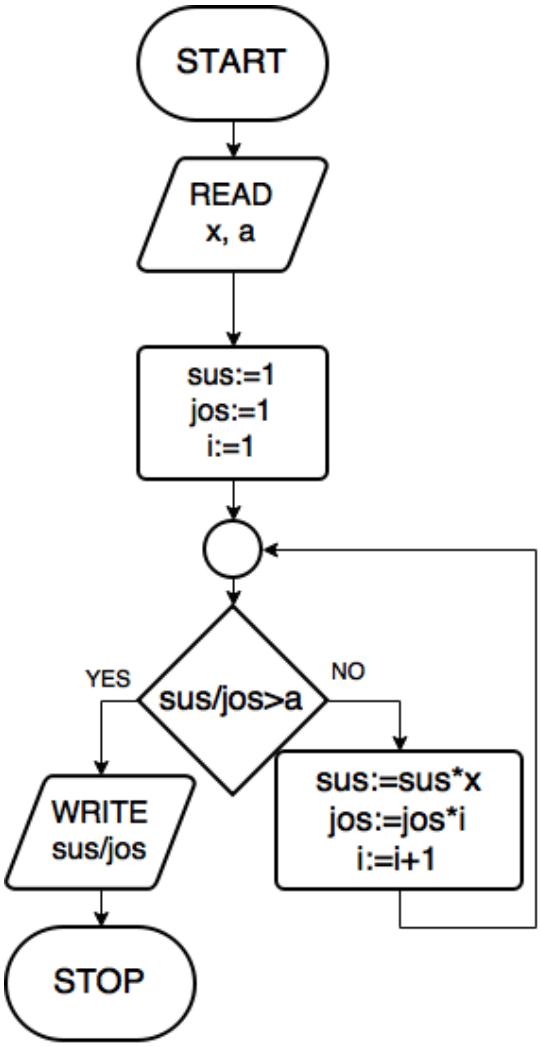


Linii de cod:

```
program suma1k;

var n,k:integer;
    s:real;
begin
{Introducere date:}
write('Introduceti N =
');readln(n);

{initialiare:}
s:=0;

{calcul:}
for k:=1 to n do
s:=s+1/k;

{afisare rezultat:}
writeln('Suma este s =',s);
readln
end.
```

48. Să se scrie un program de calcul al valorii primului element din seria funcţională infinită: $\frac{x^2}{2!}$, $\frac{x^3}{3!}$, ..., $\frac{x^n}{n!}$ a cărui valoare e mai mică decât numărul dat *a*. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

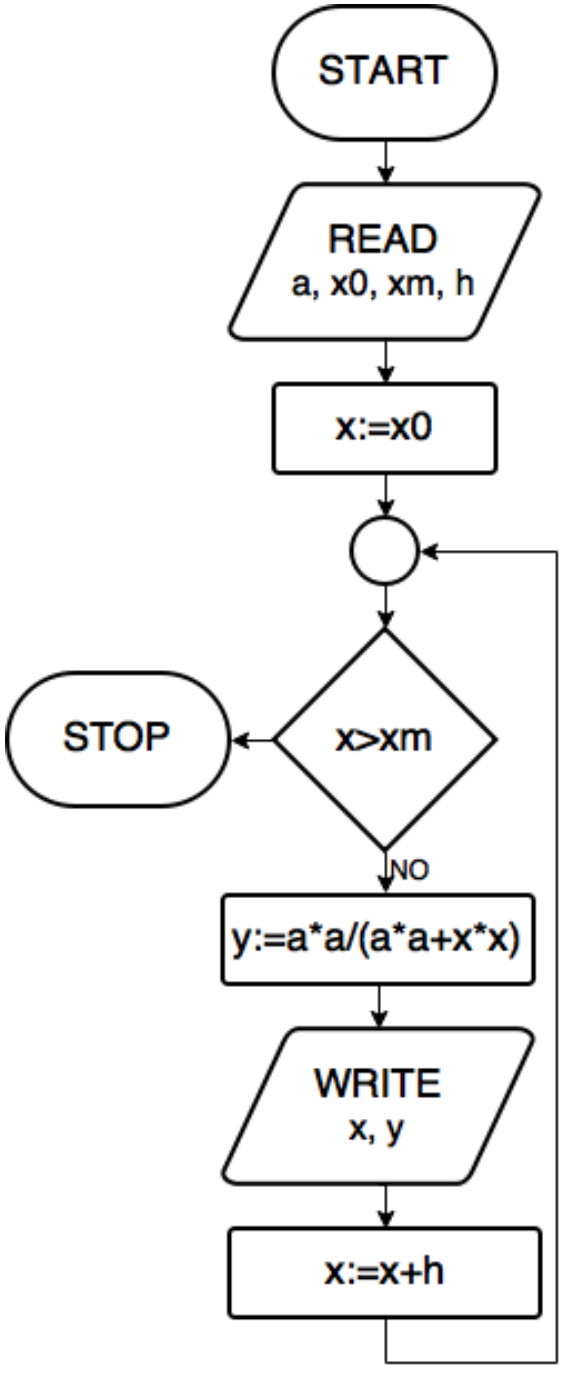


Linii de cod:

```
program serie;
var
i: word;
sus, jos, rap, a, x:real;

begin
write('Introdu x = ');
readln(x);
write('Introdu a = ');
readln(a);
sus:=1;
jos:=1;
i:=1

repeat
sus:=sus*x;
jos:=jos*i;
rap:=sus/jos;
i:=i+1
until rap<a;

writeln('Valoarea raportului
',sus,'/',jos,' < a = ',a, '
este ',rap);

end.
```

49. Să se scrie un program de calcul al valorilor funcţiei: Y = $\frac{a^3}{a^2+x^2}$, pentru argumentul x care se schimbă cu pasul 0.2 în intervalul [0,3]. Generalizare: argumentul x variază de la x_0 până la x_m cu pasul h. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

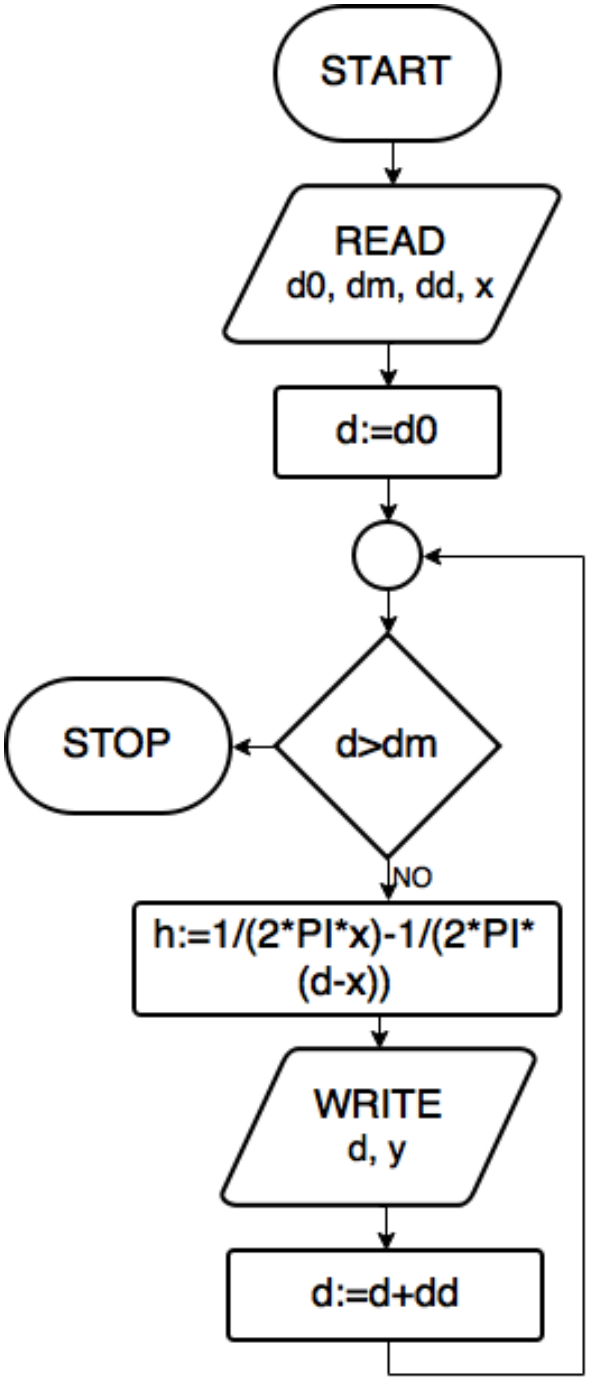


Linii de cod:

```
program pas02;
var
a, x, h, x0, xm, y: real;

begin
write('Introdu a
=');readln(a);
write('Introdu limita
inferioara x0 =');readln(x0);
repeat
write('Introdu limita
superioara xm =');readln(xm);
if xm<=x0 then
writeln('Eroare, reluati !')
until xm>x0;
repeat
write('Introdu pasul h
=');readln(h);
if h>abs(xm-x0) then
writeln('Eroare, reluati !')
until h<abs(xm-x0);

x:=x0;
repeat
y:=a*a/(a*a+x*x);
writeln('Pentru x= ',x,', Y =
',y:4:3);
x:=x+h
until x>xm;
readln;

end.
```

50. Folosind formula de calcul: $H = \frac{1}{2\pi x} - \frac{1}{2\pi(d-x)}$ realizaţi un program pentru determinarea intensităţii câmpului magnetic în punctul x amplasat între două conductoare paralele parcurse de curentul I. Distanţa d dintre conductoare variază de la d_0 până la d_m cu pasul Δd. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

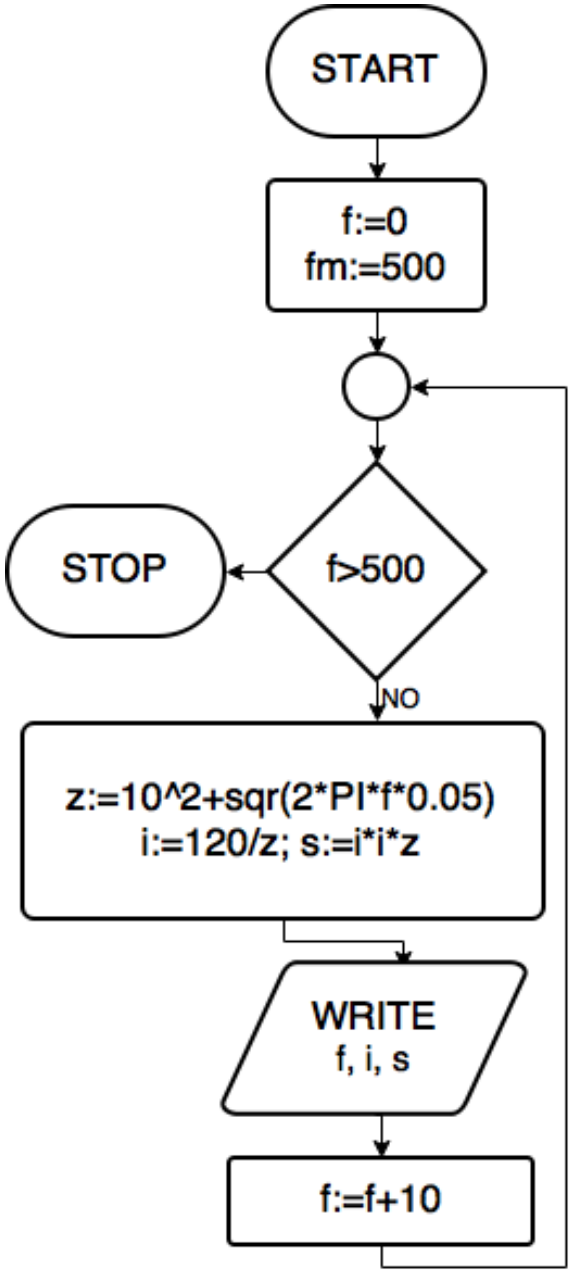


Linii de cod:

```
program camp_mag;
var d0, dm, d, h, dd, x: real;

begin

repeat
write('Introdu limita inferioara
(d0>0) =');readln(d0);
until d0>0;

repeat
write('Introdu limita superioara
dm =');readln(dm);
if dm<=d0 then writeln('Eroare,
reluati !')
until dm>d0;

repeat
write('Introdu puntul X cuprins
intre d0 si dm =');readln(x);
until (x<dm) and (x>=d0);

repeat
write('Introdu pasul delta
=');readln(dd);
if dd>abs(dm-d0) then
writeln('Eroare, reluati !')
until dd<abs(dm-d0);

d:=d0;
repeat
if x<>d then begin{in plus fata de
schema,se evita impartire la 0}
h:=1/(2*PI*x)-1/(2*PI*(d-x));
writeln('Pentru d = ',d,', Y =
',h:4:3);
end; d:=d+dd;
until d>dm;
readln; end.
```

51. O bobină cu rezistenţa activă R_e = 10 Ω, inductivitatea L = 0.05 Hn se alimentează de la o sursă de curent alternativ cu tensiunea U = 120 V şi frecvenţa *f* variabilă (f_0 = 0, f_m = 500 Hz). Să se scrie un program de calcul al curentului şi puterii absorbite de la sursă la variaţia frecvenţei f_0 până la f_m cu pasul h = 10 Hz. Formulele de calcul: I = U/Z, $Z = \sqrt{R^2 + (2\pi fl)^2}$, $S = I^2Z$. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

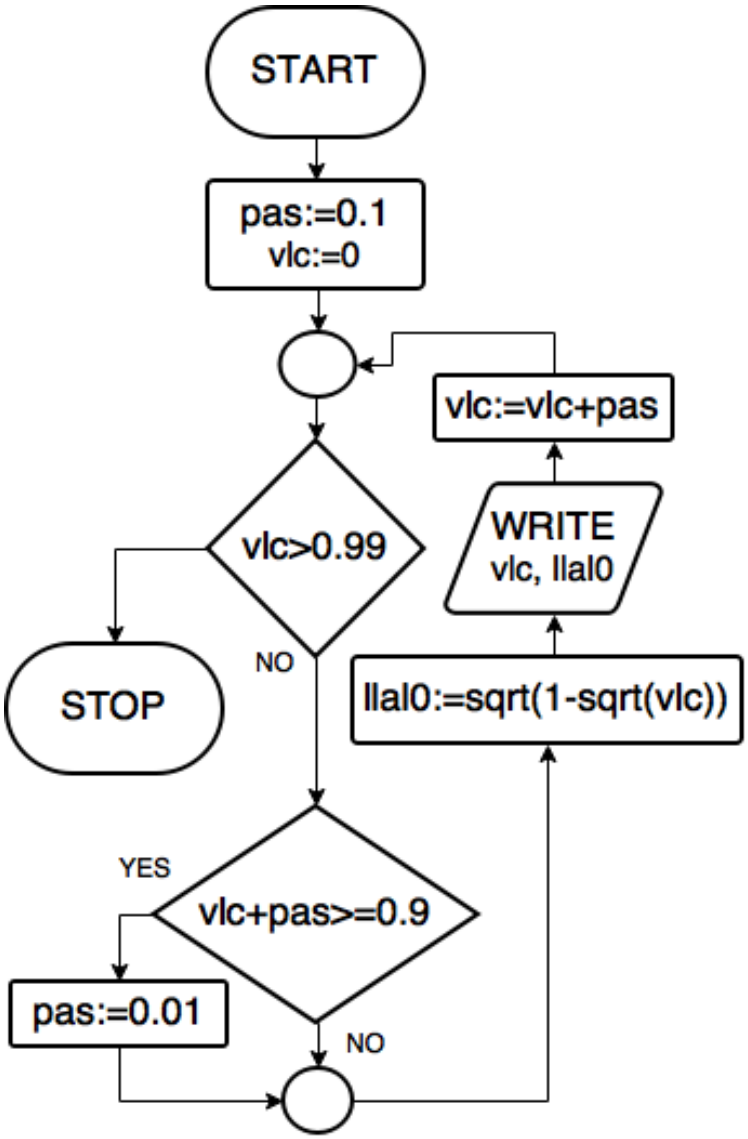


Linii de cod:

```
program putere_abs;
var
   f, f0, fm: word;
   i, z, s: real;

begin
f:=0;
repeat
z:=sqrt(10*10+sqr(2*PI*f*0.05)
);
i:=120/z;
s:=i*i*z;
writeln('Pentru f=',f,',
i=',i:4:2,' S=',s:4:2);
writeln;

f:=f+10
until f=500;

end.
```

52. În teoria relativităţii restrânse, construcţia lungimilor este dată de relaţia: $l = l_0 \cdot \sqrt{(1 - v^2 / c^2)}$, unde v este viteza corpului iar c este viteza luminii. Să se calculeze şi să se afişeze valorile raportului l/l_0, pentru valori ale raportului v/c de la 0 la 0.9 cu pasul 0.1 şi de la 0.9 la 0.99 cu pasul 0.01. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

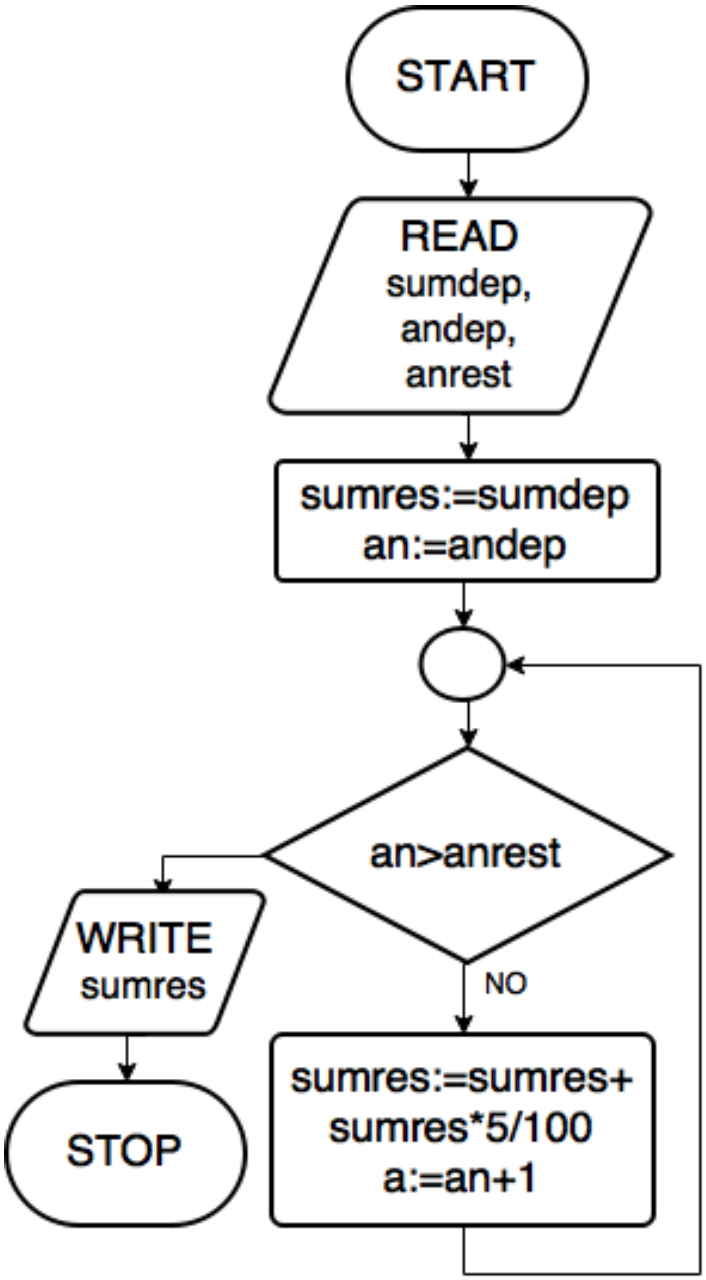


Linii de cod:

```
program relativ;
var
  llal0, vlc, pas: real;

begin
pas:=0.1;
vlc:=0;

repeat
llal0:=sqrt(1-sqr(vlac));
writeln('Pentru v/c =
',vlc,', raportul l/lo =
',llal0:4:3);
if vlc+pas>=0.9 then
pas:=0.01;
vlc:=vlc+pas;
until vlc >0.99;
readln
end.
```

53. Un bunic a depus la CEC 100 lei pe numele primului său nepoţel, născut la 1 ianuarie 2005. Presupunând că nu s-au efectuat nici depuneri, nici restituiri şi că dobânda anuală acordată a fost de 5%, să se determine care a fost suma trecută pe carnetul de economii la 1 ianuarie 2015. Se cere o generalizare, în sensul că se vor citi valori pentru suma iniţială, anul depunerii, anul pentru care se cere suma şi dobânda anuală. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

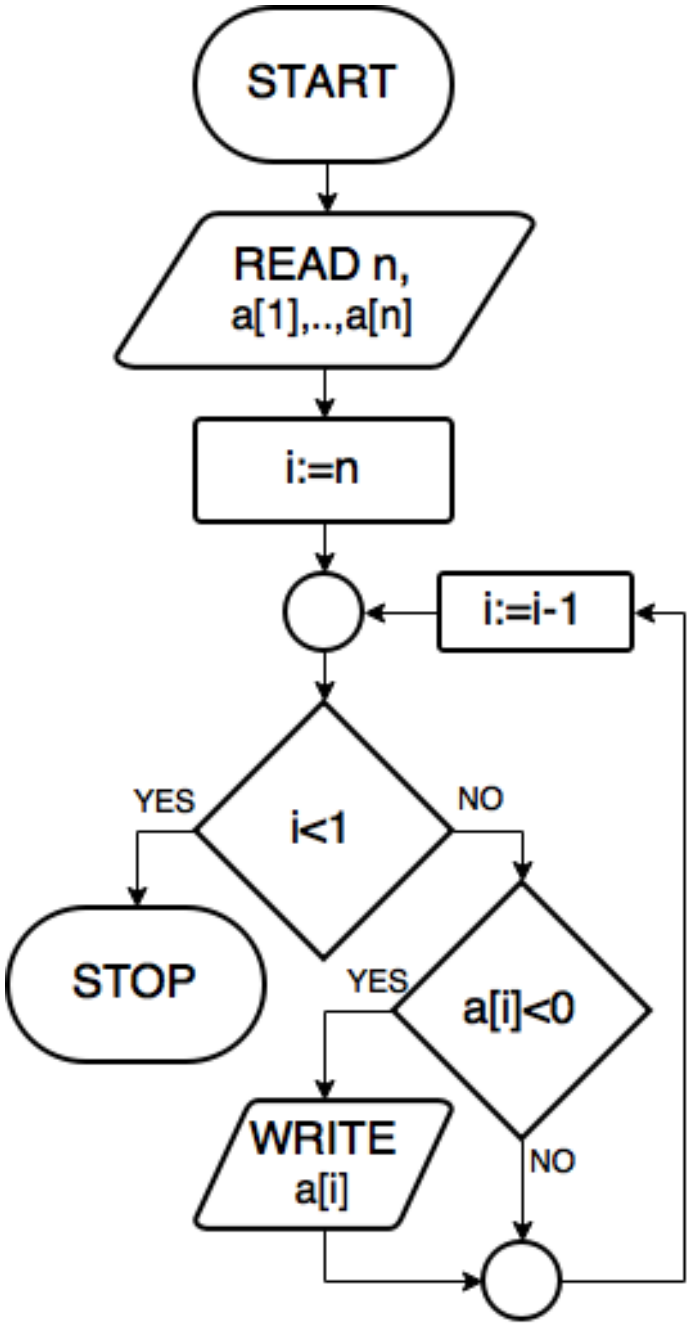


Linii de cod:

```
program buni;
var
  andep, anrest, an: word;
  sumdep, sumres: real;

begin
write('suma depusa = ');
readln(sumdep);
write('anul depunerii = ');
readln(andep);
repeat {in plus fata de
schema, evita anrest<andep}
write('anul retragerii = ');
readln(anrest);
until anrest >andep;

sumres:=sumdep;
for an:=andep to anrest do
sumres:=sumres+sumres*5/100;

writeln('Suma retrasa este =
',sumres:4:3);
readln
end.
```

54. Să se scrie un program care să afişeze în ordine inversă elementele negative ale unui şir de numere reale. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

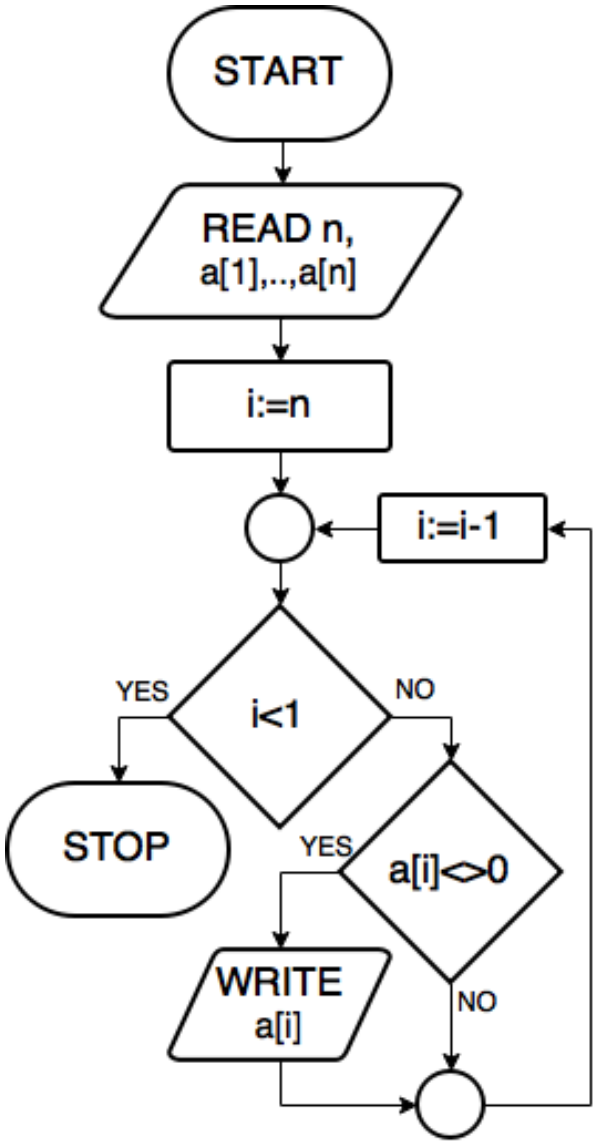


Linii de cod:

```
program inv_neg;
var
  a:array[1..100] of real;
  i, n: byte;

begin
write('Introduceti n = ');
readln(n);
  for i:=1 to n do
  begin
  write('a',i,' =
');readln(a[i]);
  end;

  writeln;
  writeln('numere
negative:');
  for i:=n downto 1 do
  if a[i]<0 then
  writeln('a',i,' = ',a[i]);
  readln
end.
```

55. Să se scrie un program care să afişeze în ordine inversă elementele diferite de zero ale unui şir de numere reale. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

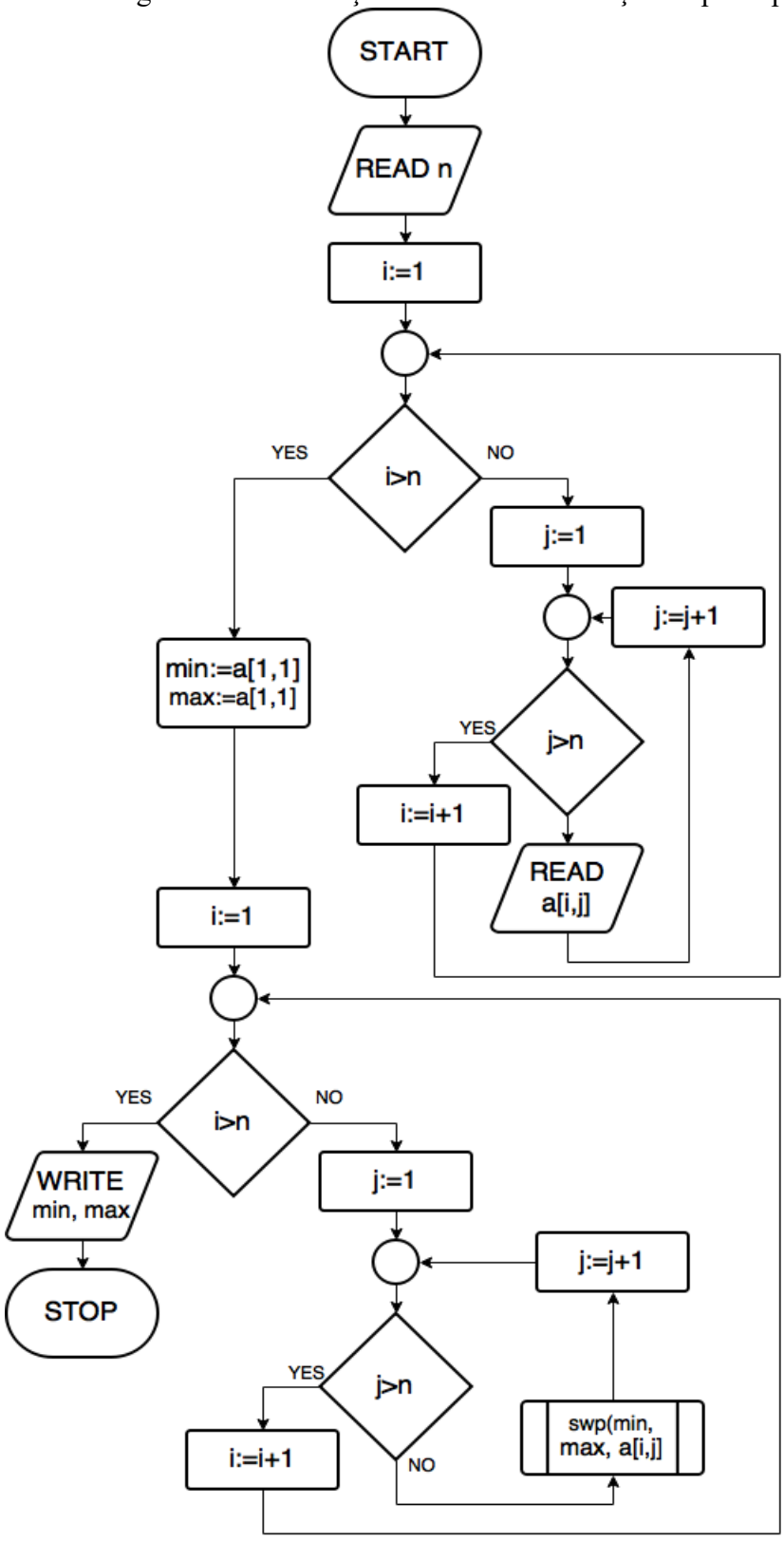
Schema logică:



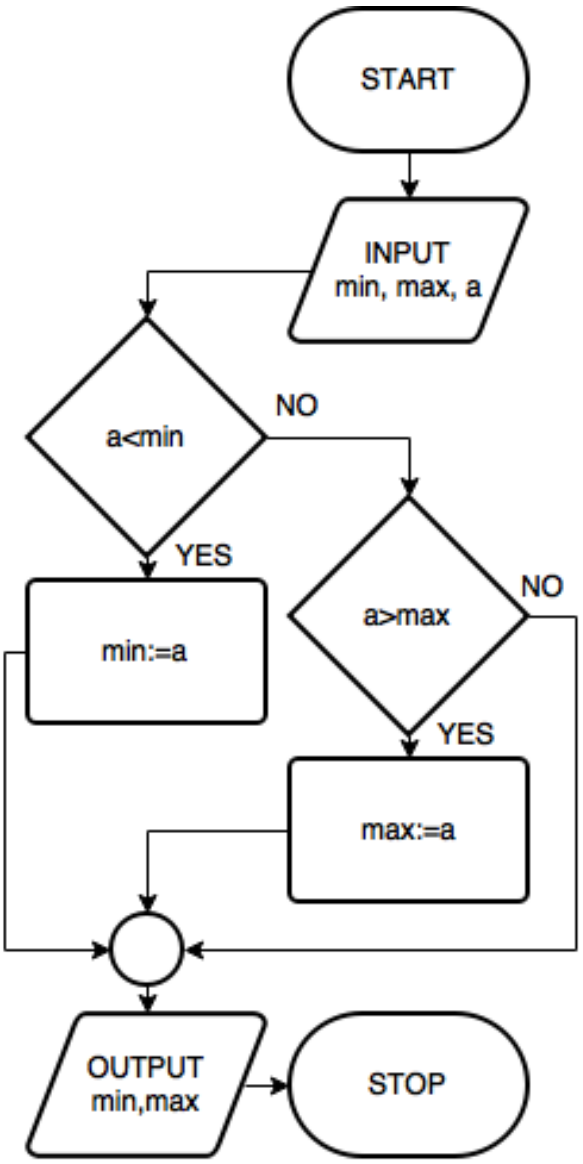
Linii de cod:

```
program inv_dif_zero;
var
  a:array[1..100] of real;
  i, n: byte;

begin
write('Introduceti n = ');
readln(n);
  for i:=1 to n do
  begin
  write('a',i,' =
');readln(a[i]);
  end;

  writeln;
  writeln('numere
negative:');
  for i:=n downto 1 do
  if a[i]<>0 then
  writeln('a',i,' = ',a[i]);
  readln
end.
```

56. Să se scrie un program care să calculeze simultan valoarea maximă şi minimă a unei matrici pătratice de numere întregi. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.



Schema logică procedura swp():



Linii de cod:

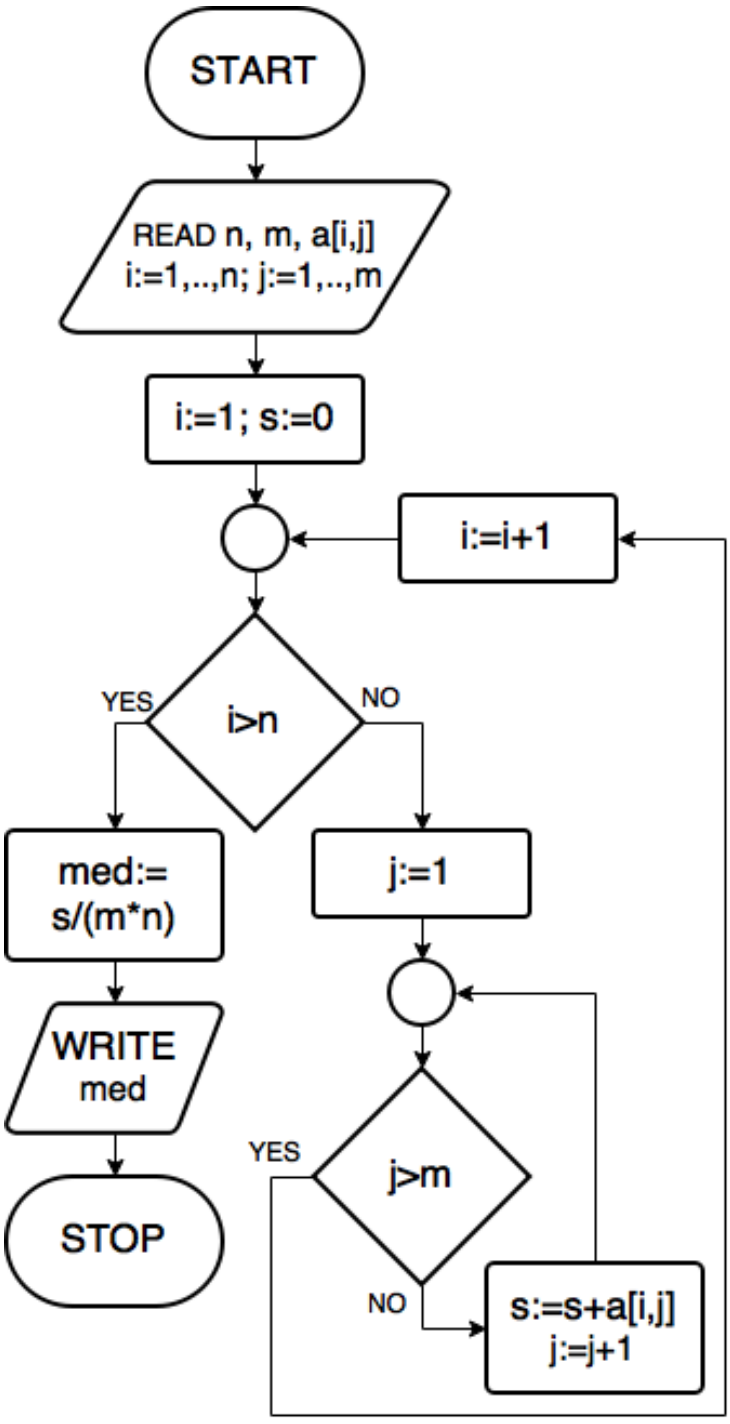
```
program min_max;
var
  a:array[1..100,1..100] of
real;
  min, max: real;
  i, j, n: byte;

begin
write('Introduceti n = ');
readln(n);
  for i:=1 to n do
  for j:=1 to n do
  begin
  write('a',i,j,' =
');readln(a[i,j]);
  end;

  min:=a[1,1]; max:=a[1,1];
  for i:=1 to n do
  for j:=1 to n do
  begin
  if a[i,j]>max
then max:=a[i,j];
 if a[i,j]<min
then min:=a[i,j]
  end;

  writeln;
  writeln('min = ',min);
  writeln('max = ',max);
  readln
end.
```

Observatie:

Liniile scrise cu caractere mai mari, fac obiectul procedurii **swp(min, max, a[i,j])** din schema logica alaturata.

57. Să se scrie un program care să afişeze media aritmetică a elementelor unei matrici de dimensiune n×m de numere întregi. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică procedura swp():

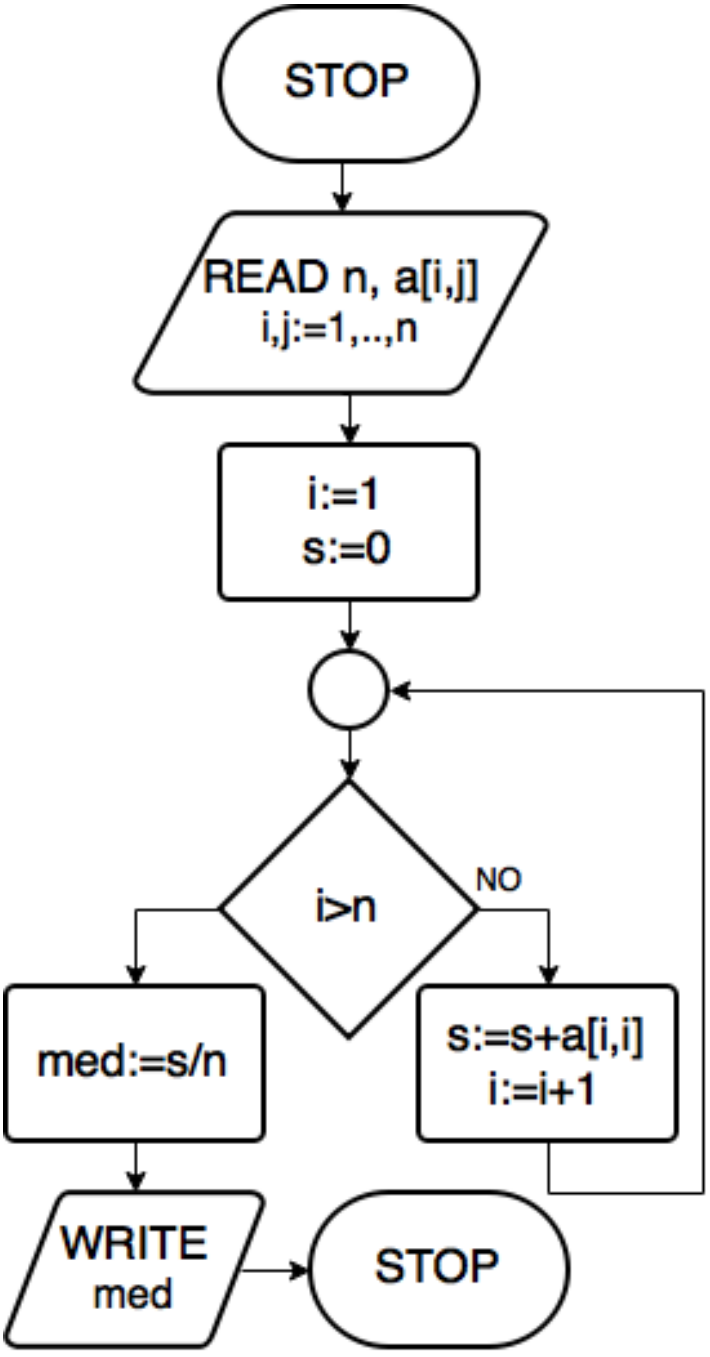


Linii de cod:

```
program media_mn;
var
  a:array[1..100,1..100] of
integer;
  med, sum: real;
  i, j, n, m: byte;

begin
write('Introduceti n = ');
readln(n);
write('Introduceti m = ');
readln(m);
  for i:=1 to n do
  for j:=1 to m do
  begin
  write('a',i,j,' = ');
readln(a[i,j]);
  end;

  sum:=0;
  for i:=1 to n do
  for j:=1 to m do
  sum:=sum+a[i,j];
  med:=sum/(m*n);

  writeln;
  writeln('media =
',med:4:3);
  readln
end.
```

58. Să se scrie un program care să afişeze media aritmetică a elementelor de pe diagonala principală a unei matrici pătratice de numere intregi. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

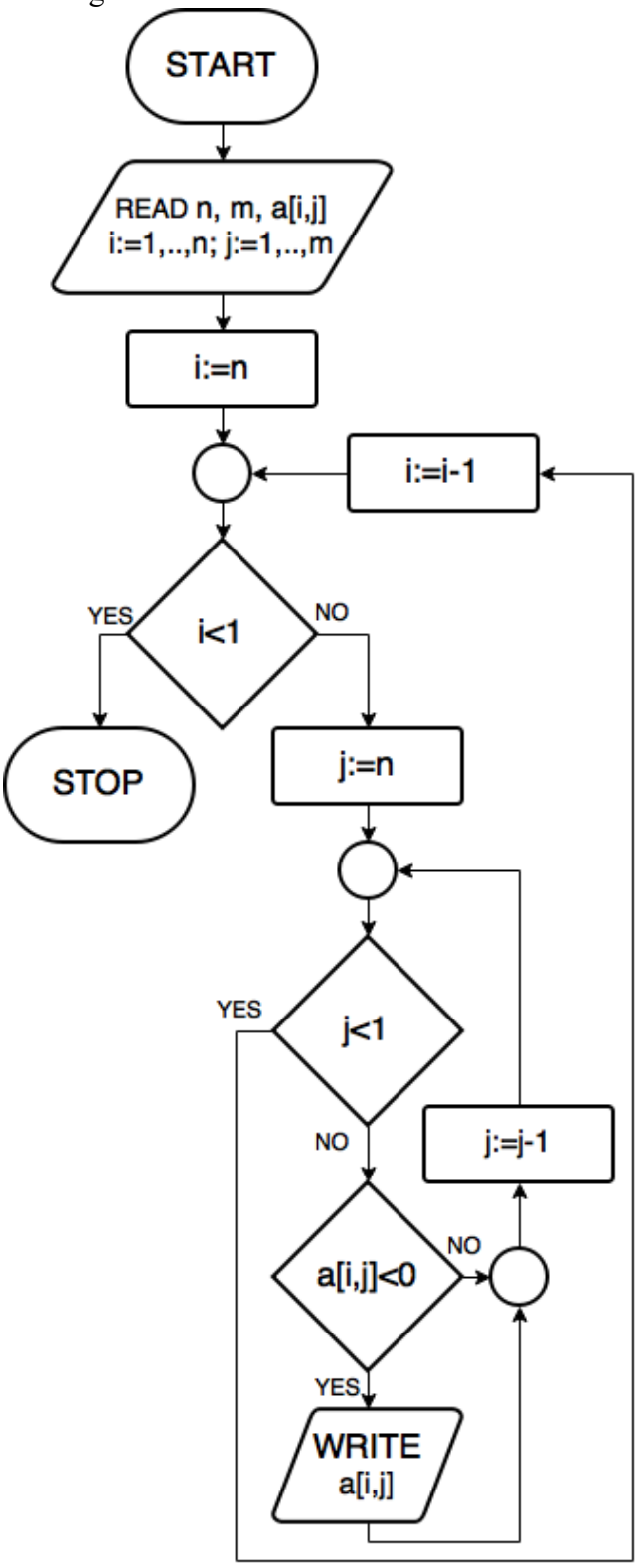


Linii de cod:

```
program media_diagp;
var
  a:array[1..100,1..100] of
integer;
  med, sum: real;
  i, j, n: byte;

begin
write('Introduceti n = ');
readln(n);
  for i:=1 to n do
  for j:=1 to n do
  begin
  write('a',i,j,' =
');readln(a[i,j]);
  end;

  sum:=0;
  for i:=1 to n do
  sum:=sum+a[i,i];
  med:=sum/n;

  writeln;
  writeln('media d. p. =
',med:4:3);
  readln
end.
```

59. Să se scrie un program care să afişeze în ordine inversă elemetele negative ale unei matrici de numere reale. Toate cererile şi rezultatele vor fi afişate explicit pe ecran.

Schema logică:

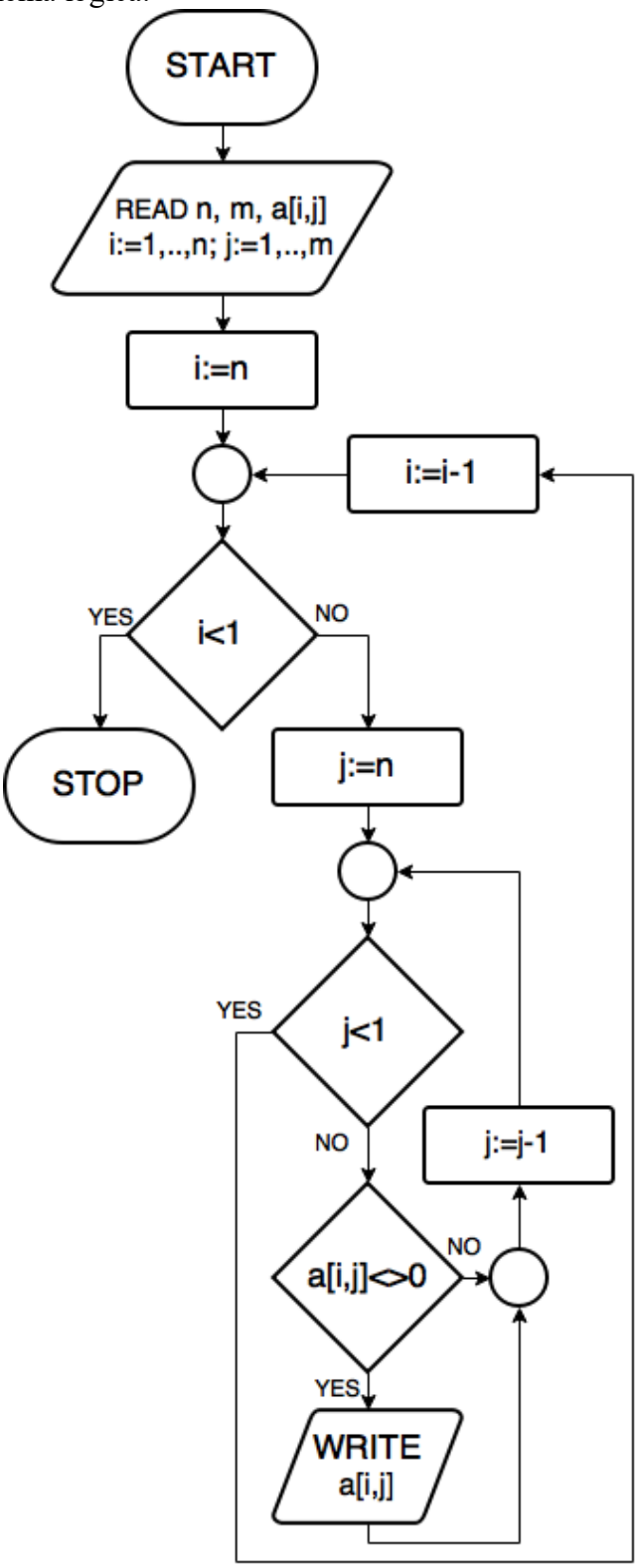


Linii de cod:

```
program inv_neg;
var
  a:array[1..100,1..100] of
real;
  i, j, m, n: byte;

begin
write('Introduceti m = ');
readln(m);
write('Introduceti n = ');
readln(n);
  for i:=1 to n do
  for j:=1 to m do
  begin
  write('a',i,j,' =
');readln(a[i,j]);
  end;

  writeln;
  writeln('numere
negative:');
  for i:=n downto 1 do
  for j:=m downto 1 do
  if a[i,j]<0 then
  writeln('a',i,j,' =
',a[i,j]);
  readln
end.
```

60. Să se scrie un program care să afişeze în ordine inversă elemetele diferite de zero ale unei matrici de numere reale.

Schema logică:



Linii de cod:

```
program inv_neg;
var
  a:array[1..100,1..100] of
real;
  i, j, m, n: byte;

begin
write('Introduceti m = ');
readln(m);
write('Introduceti n = ');
readln(n);
  for i:=1 to n do
  for j:=1 to m do
  begin
  write('a',i,j,' =
');readln(a[i,j]);
  end;

  writeln;
  writeln('Numere <> 0:');
  for i:=n downto 1 do
  for j:=m downto 1 do
  if a[i,j]<>0 then
  writeln('a',i,j,' =
',a[i,j]);
  readln
end.
```