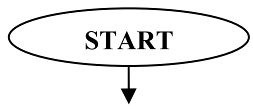
Capitolul 2 Scheme logice

2.1 Principalele elemente ale unei scheme logice

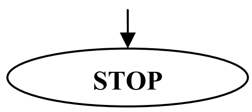
Primii paşi în domeniul programării structurate vor fi făcuţi prin intermediul schemelor logice, o metodă foarte bună pentru dezvoltarea raţionamentului. Independente de limbajul de programare, schemele logice sunt reprezentări grafice ale programelor, oferind o imagine clară asupra acestora şi permiţând o dezvoltare mai uşoară, ele fiind cel mai simplu mod de exprimare a unui algoritm. Schemele logice pot părea arhaice. Cu toate acestea, ele stau la baza conceperii unui program, fiind foarte folositoare mai ales celor care se află la început în ceea ce priveşte programarea. Se spune că o imagine valorează cât o mie de cuvinte, iar schemele logice reprezintă exact imaginea unui program, scrierea efectivă a programului, după reprezentarea lui grafică, fiind doar o muncă de rutină [GP00].

Schemele logice sunt compuse din câteva simboluri cu ajutorul cărora se poate reprezenta orice algoritm. Elementele care pot intra în componenţa unei scheme logice sunt prezentate în continuare.

Blocul START marchează începutul programului:

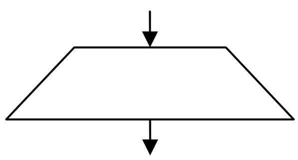


Blocul STOP marchează sfârşitul programului:

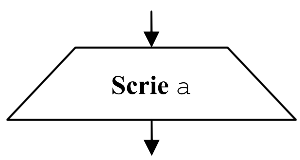


Prin intermediul acestor două blocuri se poate şti cu precizie unde începe şi unde se încheie reprezentarea unui program. Se elimină astfel orice nelămuriri privitoare la delimitarea schemelor logice.

Blocul de ieşire este folosit pentru afişarea rezultatelor:



De exemplu:



Utilizând aceste trei blocuri se poate contura schema logică a unui prim program care să tipărească mesajul „`Acesta este primul meu program`” (figura 2.1.1). Aşa cum se poate observa, schema cuprinde un bloc de ieşire (utilizat pentru tipărirea mesajului) şi cele două blocuri care marchează începutul, respectiv sfârşitul programului.

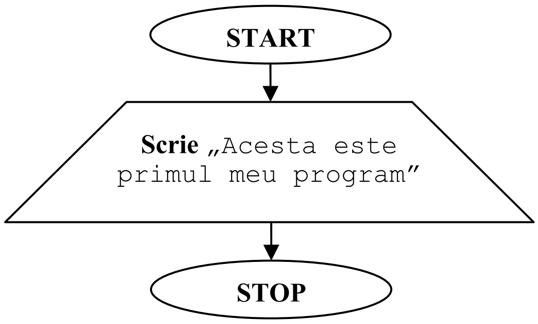
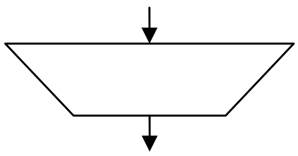
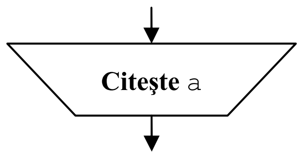


Figura 2.1.1 Schemă logică în care este utilizat doar blocul de ieşire

Blocul de intrare este folosit pentru citirea/introducerea datelor.



De exemplu:



În schema logică din figura 2.1.2 blocul de intrare este folosit pentru introducerea numelui unei persoane, nume care este apoi afişat prin intermediul unui bloc de ieşire.

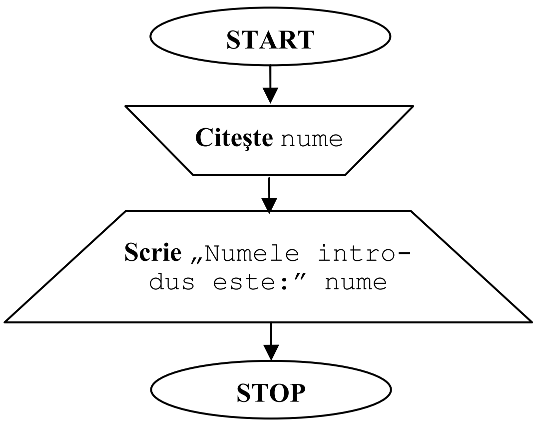
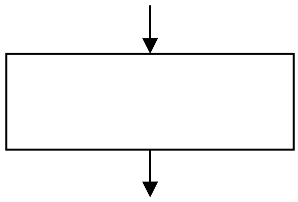
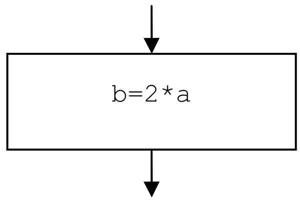


Figura 2.1.2 Schemă logică în care sunt utilizate blocurile de intrare şi de ieşire

Blocul de calcul execută operaţii şi este utilizat atunci când are loc un proces de prelucrare a datelor (calcule sau iniţializări):



De exemplu:



Cu ajutorul blocului de calcul se pot rezolva o serie de probleme. În continuare este prezentată schema logică pentru aflarea sumei şi produsului a două numere introduse de la tastatură (figura 2.1.3). Pentru citirea numerelor se foloseşte un bloc de intrare. Apoi, prin intermediul unui bloc de calcul se realizează suma şi produsul care, în final, sunt tipărite cu ajutorul unui bloc de ieşire.

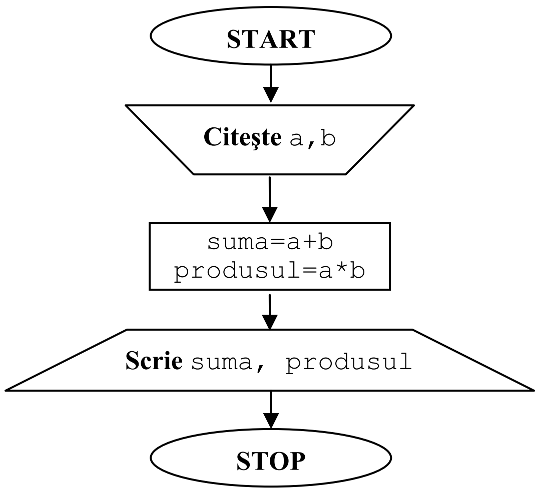


Figura 2.1.3 Schema logică pentru calcularea sumei şi produsului a două numere

Având la dispoziţie şi blocul de calcul, se pot soluţiona şi probleme ceva mai complexe. De exemplu ecuaţia de gradul întâi: `a`**`x`**`+b=0`. Se citesc de la tastatură două numere `a` şi `b`, reprezentând coeficienţii şi se calculează valoarea necunoscutei `x`, valoare care apoi este afişată. În figura 2.1.4 este prezentată o primă schemă logică pentru rezolvarea ecuaţiei de gradul întâi.

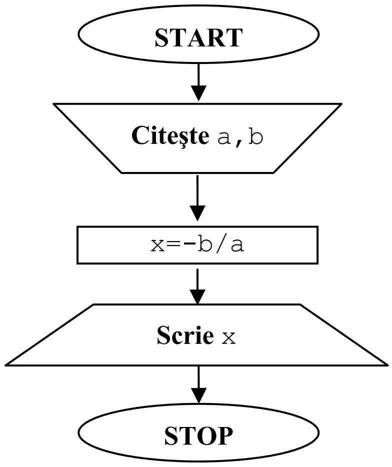


Figura 2.1.4 Schema logică (incompletă) pentru rezolvarea ecuaţiei de gradul întâi

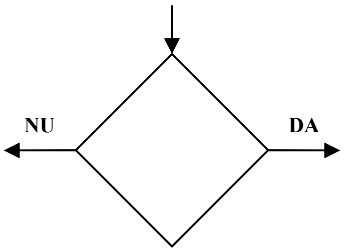
Aşa cum se poate observa, această schemă realizată pentru rezolvarea ecuaţiei de gradul întâi are un neajuns: nu tratează situaţia în care coeficientul `a` este egal cu zero (caz în care împărţirea lui `b` la `a` nu se mai poate executa). Din punct de vedere matematic o ecuaţie de forma `a`**x**`+b=0` se rezolvă astfel:

Se verifică valoarea coeficientului `a`:

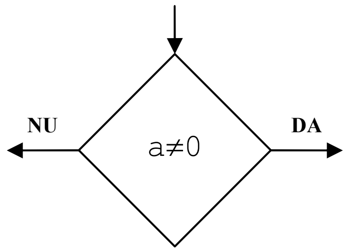
- dacă `a` e diferit de zero, atunci `x=-b/a;`
- dacă nu, atunci ecuaţia devine `b=0` şi trebuie verificată valoarea coeficientului `b`:
    - dacă `b` e diferit de zero, atunci expresia `b=0` e falsă, deci nu există soluţii pentru necunoscuta `x`;
    - dacă `b` este egal cu zero, atunci expresia `b=0` e adevărată şi orice număr real reprezintă o soluţie a ecuaţiei considerate (adică necunoscuta `x` are o infinitate de soluţii).

Pentru a rezolva însă această problemă, mai e nevoie de un bloc: cel prezentat în continuare.

Blocul decizional evaluează condiţii şi este utilizat atunci când într-un program trebuie să se ia o decizie bazată pe două alternative. Blocul trebuie să conţină o întrebare la care să se poată răspunde prin „da” sau „nu”:



De exemplu:



Folosind şi blocul decizional, se poate rezolva, corect, ecuaţia de gradul întâi conform algoritmului descris anterior. Schema logică prezentată în figura 2.1.5 urmează întocmai paşii acestui algoritm.

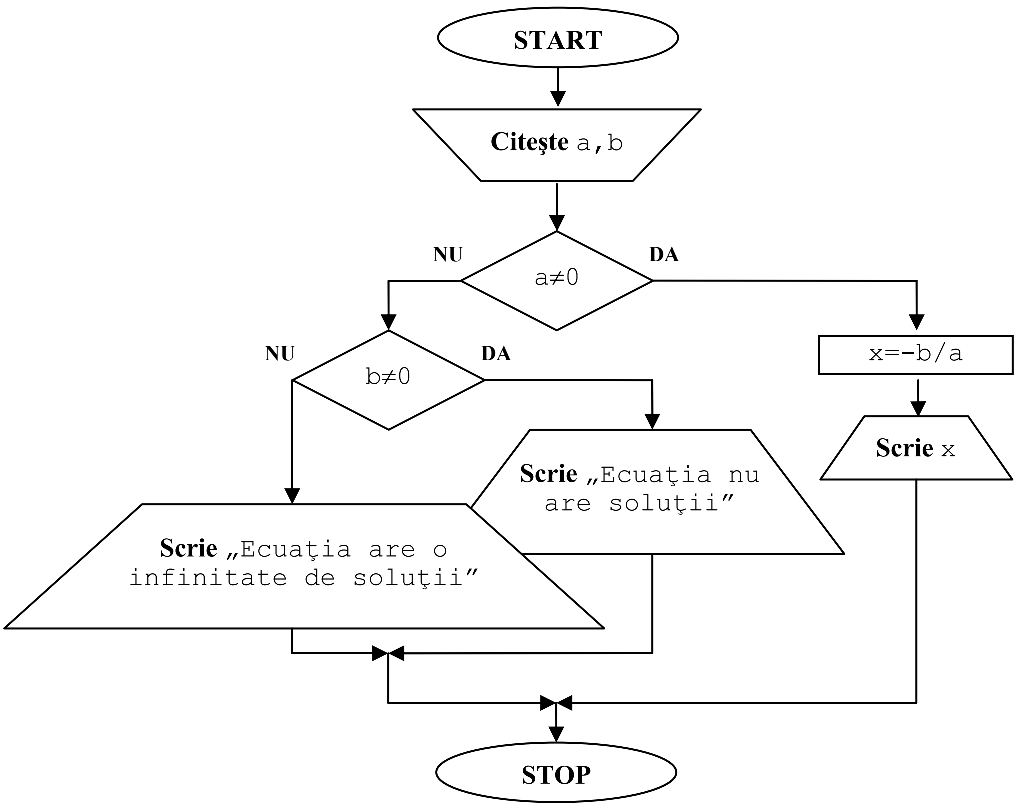


Figura 2.1.5 Schema logică pentru rezolvarea ecuaţiei de gradul întâi

2.2 Probleme rezolvate

Schemele logice sunt o modalitate simplă şi clară de reprezentare a unui algoritm şi de rezolvare a unei probleme, oferind celor care fac primii paşi în programarea structurată o excelentă unealtă prin care îşi pot îmbunătăţii modul în care gândesc soluţionarea unei probleme. În continuare, pentru a fi înţelese cât mai bine noţiunile privitoare la schemele logice, se vor rezolva sau, mai bine spus, se vor propune rezolvări pentru alte câteva probleme.

2.2.1 Pentru început se doreşte calcularea ariei unui pătrat cu latura `L` dată. Această arie va fi aflată utilizând formula `Aria=L*L`. O primă schemă logică pentru rezolvarea acestei probleme este redată în figura 2.2.1. Utilizatorul trebuie să introducă (prin intermediul unui bloc de intrare) latura pătratului. Se calculează aria, care (utilizând un bloc de ieşire) este apoi afişată.

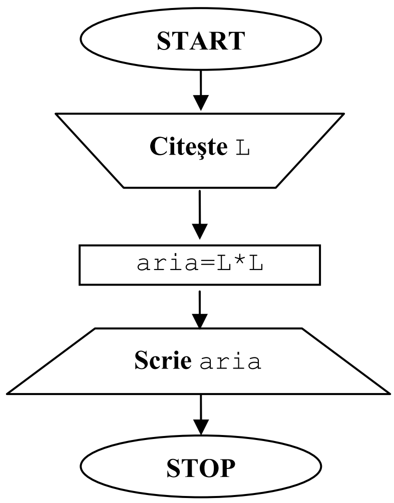


Figura 2.2.1 Aria unui pătrat – varianta 1

Ce se întâmplă însă dacă utilizatorul, din greşeală sau din neştiinţă, introduce un număr negativ pentru latura pătratului? Valoarea nu poate fi acceptată deoarece o dimensiune nu poate fi negativă. Prin urmare schema logică din figura 2.2.1 nu funcţionează corect pentru toate cazurile şi trebuie modificată. Problema ar putea fi parţial soluţionată dacă, înainte de a fi calculată aria, s-ar testa valoarea lui `L`. În cazul în care latura este strict pozitivă, se poate calcula şi afişa aria; altfel, se va afişa un mesaj de eroare (figura 2.2.2).

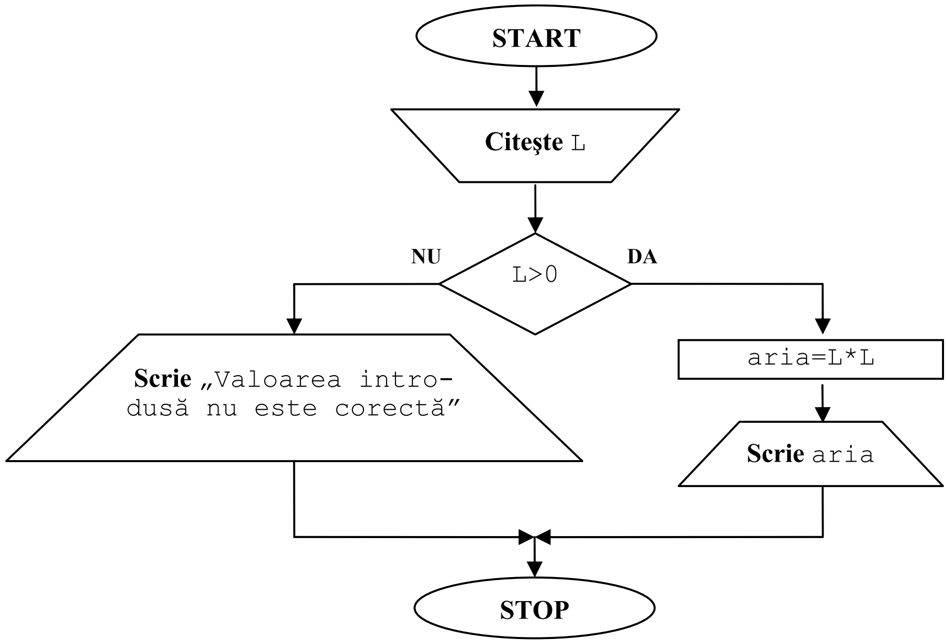


Figura 2.2.2 Aria unui pătrat – varianta 2

Există însă situaţii în care aria (sau ceva similar) trebuie neapărat calculată. Cu alte cuvinte utilizatorul trebuie forţat să introducă o valoare validă (în cazul de faţă, o valoare pozitivă). Acest lucru se realizează prin intermediul unei structuri repetitive (figura 2.2.3). Utilizatorul va trebui să reintroducă valoarea cerută atâta timp cât aceasta nu este validă. Aşadar, se va trece mai departe, calculându-se aria, numai după ce este introdusă o valoare pozitivă pentru latură.

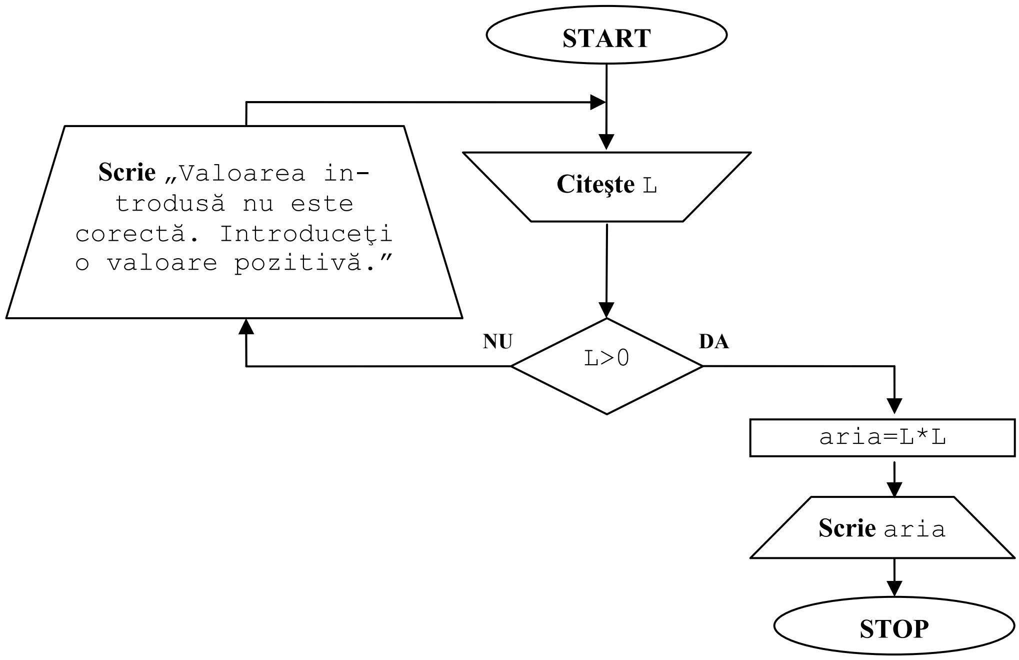


Figura 2.2.3 Aria unui pătrat – varianta 3 (completă)

2.2.2 Următoarea problemă este tot un exemplu de structură repetitivă. Utilizatorul introduce `n` numere de la tastatură (unde `n` este număr natural). Se va calcula şi se va afişa media aritmetică a acestor numere (figura 2.2.4). Pentru a putea rezolva această problemă, `n` trebuie să fie un număr pozitiv. Se va folosi în acest sens o structură repetitivă de validare a citirii lui `n` asemănătoare cu cea de la calculul ariei unui pătrat de latură `L`. După ce se citeşte un număr `n` pozitiv, se poate trece la calculul mediei aritmetice. Aceasta reprezintă suma elementelor introduse raportată la numărul de elemente. Prin urmare se iniţializează cu zero o variabilă numită `suma` (iniţial, adică înainte de a se introduce vreun număr, suma este evident zero). De asemenea, în orice moment al execuţiei programului, trebuie ştiut câte numere au fost deja citite. Pentru acest lucru se foloseşte o altă variabilă, numită `contor`, şi iniţializată tot cu valoarea zero. Se poate trece apoi la citirea celor `n` numere. Acest lucru se realizează într-o structură repetitivă. De fiecare dată când un număr este citit, el se adună la sumă, iar contorul se măreşte cu o unitate. Această buclă se repetă atâta timp cât contorul este mai mic decât `n`. În momentul în care condiţia nu mai este îndeplinită, se iese din structura repetitivă şi se continuă execuţia programului cu operaţia de calculare a mediei aritmetice, care apoi poate fi afişată.

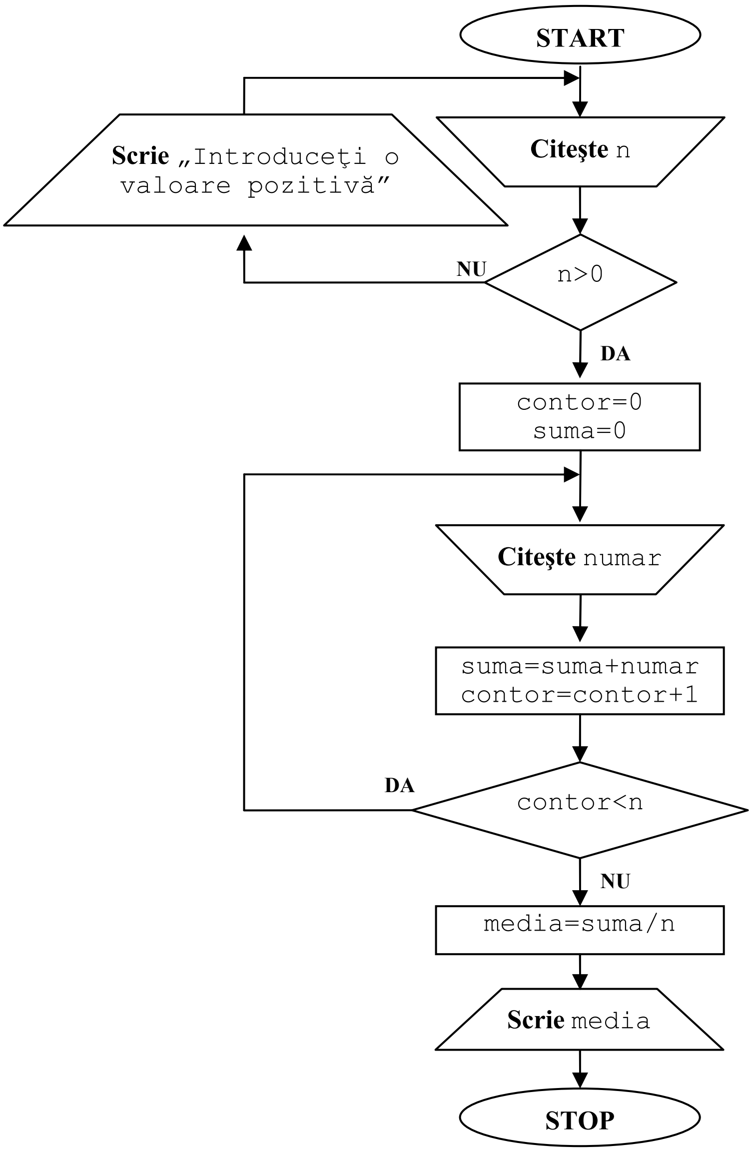
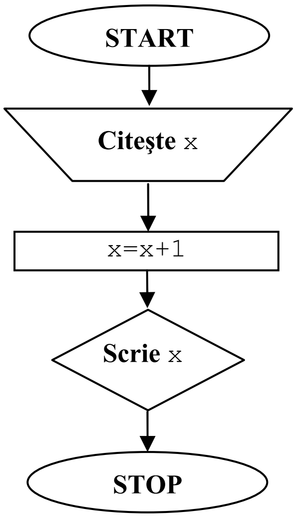


Figura 2.2.4 Media aritmetică a n numere

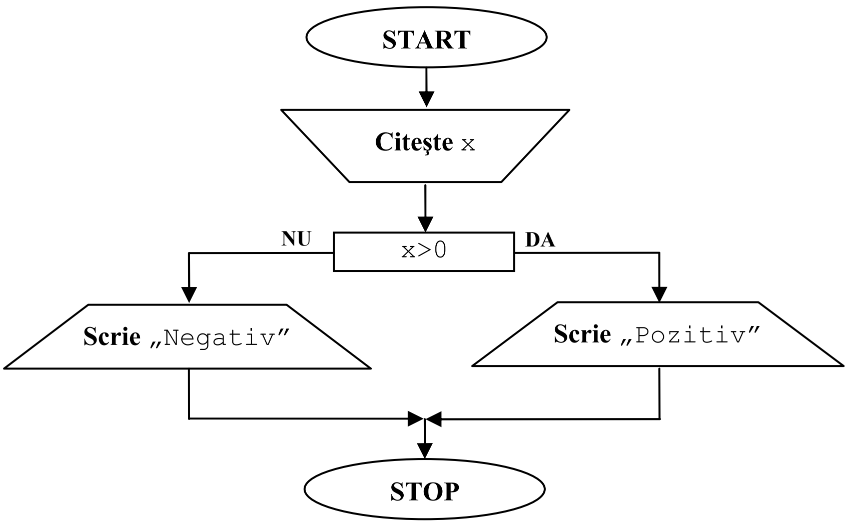
2.3 Probleme propuse

A. Specificaţi unde este greşeala.

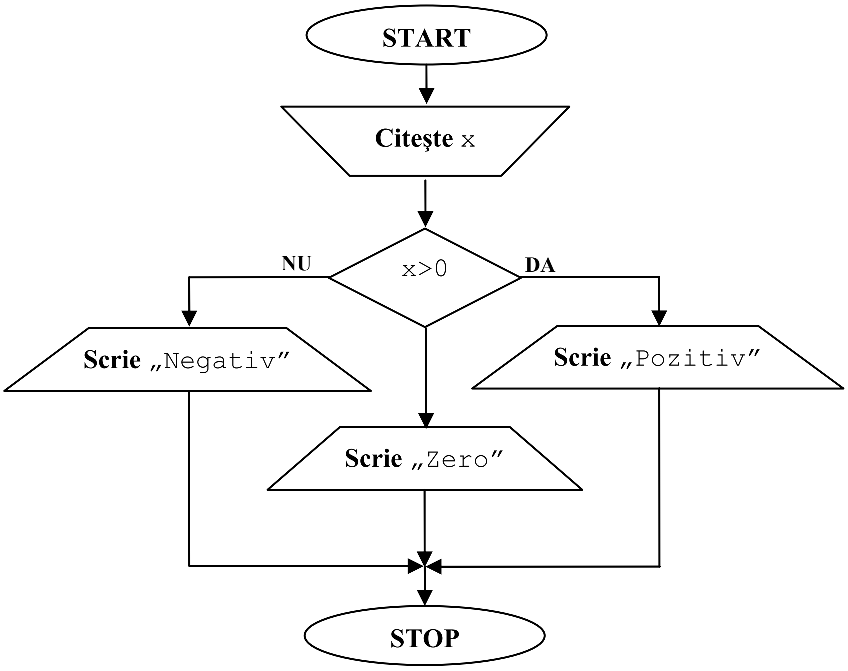
1.



2.

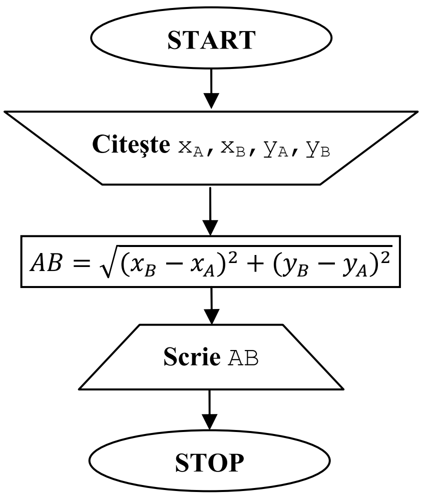


3.

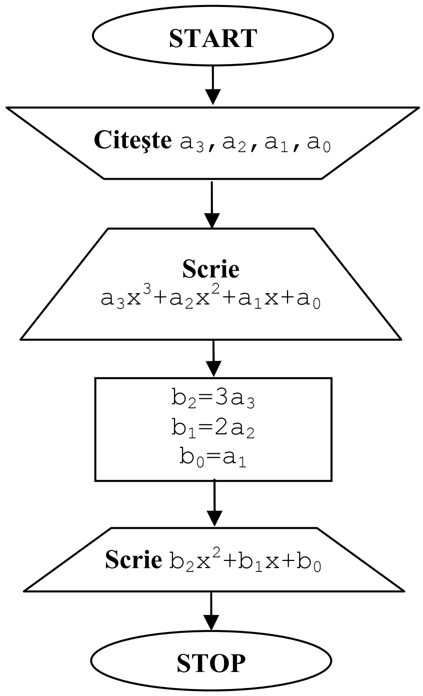


B. Se consideră următoarele scheme logice. Specificaţi ce acţiuni realizează acestea.

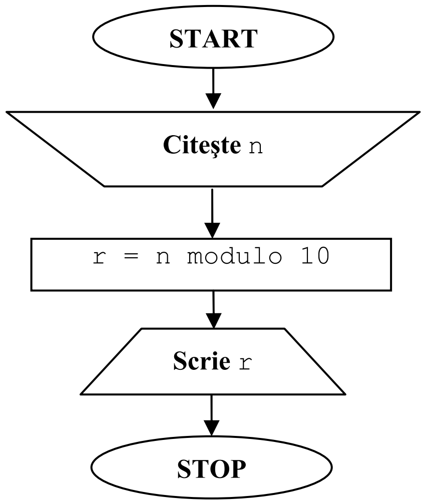
4.



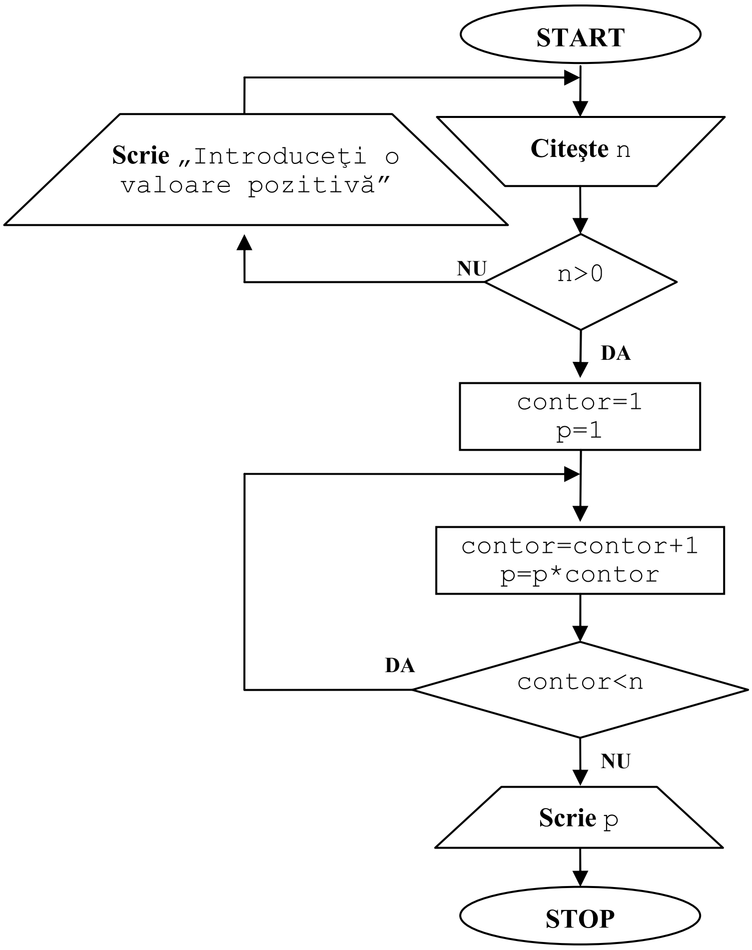
5.


START
Citeşte a_3, a_2, a_1, a_0
Scrie
$a_3x^3+a_2x^2+a_1x+a_0$
$b_2=3a_3$
$b_1=2a_2$
$b_0=a_1$
Scrie $b_2x^2+b_1x+b_0$
STOP

6.


START
Citeşte n
r = n modulo 10
Scrie r
STOP

7.



C. Alegeţi răspunsul corect (unul singur).

8. Care dintre următoarele afirmaţii referitoare la schemele logice este adevărată?

□ a) sunt reprezentări grafice ale programelor; □ b) sunt independente de limbajul de programare; □ c) reprezintă o metodă foarte bună pentru dezvoltarea raţionamentului; □ d) oferă o imagine clară asupra programelor; □ e) toate răspunsurile sunt corecte.

9. Care dintre următoarele blocuri nu există?

□ a) blocul decizional; □ b) blocul de ieşire; □ c) blocul repetitiv; □ d) blocul de calcul; □ e) blocul de intrare.

10. Blocul decizional are formă de:

□ a) triunghi; □ b) romb; □ c) dreptunghi; □ d) trapez; □ e) stea.

11. Care dintre următoarele forme geometrice nu apar într-o schemă logică?

    □ a) dreptunghi; □ b) romb; □ c) trapez; □ d) triunghi; □ e) elipsă.

12. Blocul de calcul are formă de:

    □ a) dreptunghi; □ b) cerc; □ c) romb; □ d) arc de cerc; □ e) parabolă.

13. Câte ieşiri are un bloc decizional?

    □ a) niciuna; □ b) una: `Da`; □ c) două: `Da` şi `Nu`; □ d) trei: `Da`, `Nu`, `Unknown`; □ e) oricâte, în funcţie de expresia evaluată.

D. Să se realizeze scheme logice pentru rezolvarea următoarelor probleme:

14. Se citeşte un număr real `x`. Se cere calcularea şi afişarea valorii expresiei:

    $$E = \begin{cases} \sqrt{x}, & if\ x \geq 0; \\ |x|, & if\ x < 0. \end{cases}$$

15. Se citesc trei numere `a`, `b` şi `c`, reprezentând coeficienţii ecuaţiei de gradul doi $a\mathbf{x}^2+b\mathbf{x}+c=0$. Să se găsească şi să se afişeze soluţiile acestei ecuaţii.

    Indicaţie: Din punct de vedere matematic rezolvarea ecuaţiei de gradul doi presupune mai mulţi paşi. Mai întâi se verifică valoarea coeficientului `a`:

    - dacă `a` e diferit de zero, atunci se calculează $\Delta=b^2-4ac$ şi se verifică valoarea obţinută:
        - dacă $\Delta<0$, ecuaţia nu are soluţii reale şi se afişează un mesaj în acest sens;
        - dacă $\Delta=0$, atunci ecuaţia are două soluţii egale:

          $x_1 = x_2 = \frac{-b}{2a}$;
        - dacă $\Delta>0$, ecuaţia are două soluţii distincte:

          $x_1 = \frac{-b+\sqrt{\Delta}}{2a}$ şi $x_2 = \frac{-b-\sqrt{\Delta}}{2a}$.
    - dacă `a` este egal cu zero, atunci ecuaţia devine $b\mathbf{x}+c=0$ şi trebuie verificată valoarea coeficientului `b`:

- dacă `b` este diferit de zero, atunci `x=-c/b`;
- dacă nu, atunci ecuaţia devine `c=0` şi trebuie verificată valoarea coeficientului `c`:
  - dacă `c` e diferit de zero, atunci expresia `c=0` e falsă, deci nu există soluţii pentru necunoscuta `x`;
  - dacă `c` este egal cu zero, atunci expresia `c=0` e adevărată şi orice număr real reprezintă o soluţie a ecuaţiei considerate (adică necunoscuta `x` are o infinitate de soluţii).

16. Să se calculeze aria şi lungimea unui cerc de rază `R`.

    Indicaţie: Introducerea razei trebuie validată (raza trebuie să fie pozitivă). Se vor folosi următoarele formule: aria=πR^2, iar lungimea=$2\pi R$.

17. Să se afişeze primele `n` numere ale şirului lui Fibonacci.

    Indicaţie: Şirul lui Fibonacci este un exemplu în care se foloseşte structura repetitivă. Şirul este reprezentat de o secvenţă de numere definită de următoarea relaţie:

$$F(n) = \begin{cases} 0, & if\ n = 1; \\ 1, & if\ n = 2; \\ F(n-1) + F(n-2), & if\ n > 2. \end{cases}, \quad where\ n = 1, 2, 3, \ldots$$

Cu alte cuvinte, primele două numere se dau, ele fiind `0` şi `1`, iar fiecare din următoarele numere se calculează ca sumă a precedentelor două numere. Astfel, primele `10` numere din şirul lui Fibonacci sunt:

`0, 1, 1, 2, 3, 5, 8, 13, 21, 34.`