MINISTERUL EDUCAȚIEI ȘI CERCETĂRII
ROMÂNIA
UNIVERSITATEA DE MEDICINĂ,
FARMACIE, ȘTIINȚE ȘI TEHNOLOGIE
„GEORGE EMIL PALADE"
DIN TÂRGU MUREȘ

FACULTATEA DE INGINERIE ȘI TEHNOLOGIA INFORMAȚIEI

Algoritmi fundamentali

Laborator 1

dr. ing. Kiss István

istvan.kiss@umfst.ro

Curs + laborator (seminar)

- 2 ore
  - Se prezintă teoria de bază și se explică cu ajutorul problemelor practice
- 2 ore seminar/laborator // 3 grupe
  - Se rezolvă probleme împreună cu studenții
  - Se discută eventualele aspecte de rezolvare
  - Se implementeaza algoritmii elementari cu ajutorul limbajelor si mediilor de programare
    - Skratch, AppInventor, C (Online sau Codeblocks)
- **In sistem modular 4 ore curs si 4 ore laborator pe saptamana**

Nota finală

- 50% examen din materia predată la curs
- 50% nota de laborator + activitate laborator

- Absențe laborator
  - Max 2-3, cazuri motivate
  - 4 sau peste 4 absențe: se recuperează în ultemele 2 saptamani
- Nota laborator
  - 1 sau 2 teste + activitate la laborator

Conținut laborator

1. Introducere în algoritmi.
2. Pseudocod. Bazele logicii binare.
3. Pseudocod. Algoritmi cu ramificații.
4. Pseudocod. Algoritmi ciclici.
5. Tehnici pentru elaborarea algoritmilor. Subalgoritmi.
6. Analiza algoritmilor.
7. Recursivitate.
8. Algoritmi de sortare. Tablouri unidimensionale și bidimensionale. Algoritmi elementari (prin selecție, metoda bulelor, prin inserție, sortare shell)
9. Algoritmi de sortare. Algoritm de sortare rapidă. Sortare radix. Sortare prin interclasare și prin ansambluri (heapsort).
10. Analiza algoritmilor de sortare. Comparație prin complexitate.
11. Algoritmi de căutare. Căutare secvențială. Căutare binară. Ștergere element.
12. Operații pe șiruri de caractere. Probleme.
13. Algoritmi matematici. Generare de numere aleatorii. Algoritm pentru calculul de cel mai mare divizor comun a unui număr. Numere prime. Operații cu matrici. Integrare numerică.
14. Recapitularea noțiunilor studiate. Rezolvarea unor probleme similare cu cele de la examen.

Conținut laborator - fisa disciplinei

- **Scrierea primului algoritm. Discuții despre avantajele rezolvării algoritmice de probleme. Exersarea gândirii algoritmice.**
- **Exersarea operațiilor de logică binară. Declarare date intrare-ieșire. Descrierea algoritmilor discutate la curs folosind schemă logică și pseudocod. Prezentarea mediilor de dezvoltare folosite la laborator pentru implementare și ilustrare practică (Scratch și AppInventor, Thunkable). Transpunerea algoritmilor din pseudocod în limbajul de programare C.**
- **Scriere de subalgoritmi. Asamblarea subalgoritmilor în rezolvarea problemelor. Aplicarea tehnicilor de programare ascendentă și descendentă.**
- **Exerciții de analiză a complexității algoritmilor. Evaluarea timpului de execuție.**
- **Rezolvări de probleme cu algoritmi recursivi. Aplicarea paradigmei Divide-et-Impera. Eliminarea recursivității. Problema turnurilor din Hanoi.**
- **Citirea și afișarea tablourilor uni și bidimensionale. Aplicarea algoritmilor de sortare pe șiruri de numere. Aplicații practice.**
- **Determinarea complexității algoritmilor de sortare. Evidențierea avantajelor și dezavantajelor.**
- **Aplicarea algoritmilor de căutare pe șiruri de numere. Căutare secvențială și căutare binară. Exerciții de ștergere element dintr-un tablou. Analiza complexității algoritmilor.**
- **Implementare algoritmică de operații pe șiruri de caractere. Scriere și aplicare de subprograme de prelucrare (lungime, copiere, concatenare, comparare, inserare, căutare, ștergere, tokenizare) și de conversie (din șir în număr și invers).**
- **Algoritmi matematici. Generare de numere aleatorii. Algoritm pentru calculul de cel mai mare divizor comun a unui număr. Numere prime. Operații cu matrici. Integrare numerică.**
- **Recapitularea noțiunilor studiate. Rezolvarea unor probleme similare cu cele de la examen.**

Bibliografie laborator

- [1] *** - *MIT App Inventor*. Disponibil Online: http://appinventor.mit.edu/explore/
- [2] *** - Getting started with Scratch. Disponibil Online: https://scratch.mit.edu/
- [3] Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. – *The scratch programming language and environment*. ACM Trans. Comput. Educ. 10, 4, Article 16 (November 2010).
- [4] *** - *MIT Scratch*. Disponibil Online: https://scratch.mit.edu/
- [5] *** - Online compiler and debugger for C/C++. Disponibil online: https://www.onlinegdb.com/online_c_compiler

Cuprins primul laborator

1. Scriere primul algoritm
2. Exersarea gândirii algoritmice

1. Un alt algoritm – Adunarea a două numere

- Problemele trebuie rezolvate astfel încât rezolvarea să poate fi transpusă într-un program de calculator:
- Algoritm adunare:
  - Se citește primul număr de pe tastatură
  - Se citește al doilea număr de pe tastatură
  - Se efectuază operația de adunare
  - Se salvează rezultatul într-un număr rezultat
  - Se afișează pe ecran numărul rezultat

1. Un alt algoritm – Adunarea a două numere

- Problemele trebuie rezolvate astfel încât rezolvarea să poate fi transpusă într-un program de calculator:
- Algoritm adunare:
  - Algoritm adunare este

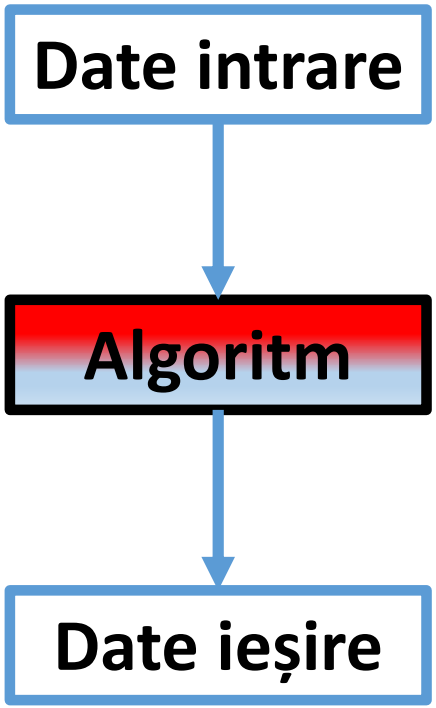
    Intreg a,b,c; - *date intrare și ieșire*

    Citeste a;

    Citeste b;

    c:=a+b;

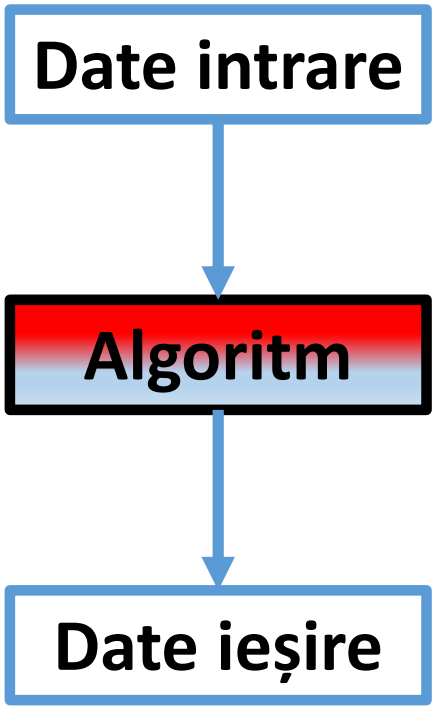
    Afișează c;
  - Sfarsit algoritm



1. Un alt algoritm – Adunarea a două numere

- Problemele trebuie rezolvate astfel încât rezolvarea să poate fi transpusă într-un program de calculator:
- Program în C pentru adunare:

```
#include<stdio.h>
int main(){
        int a,b,c;
        scanf("%d",&a);
        scanf("%d",&b);
        c = a + b;
        printf("%d",c);
        return 0;
}
```



3. Gândire algoritmică

- Se ia în considerare interdependența:
  Sistem de calcul – Algoritmică – Programare

- Necesită:
  - Cunoștințe clare de metode și tehnici
  - Capacitate de sinteză
  - Capacitate de creație
  - Capacitatea spargere a problemelor complexe în subprobleme simple
- *Algoritmizarea este cheia rezolvării problemelor cu ajutorul sistemelor de calcul*

3. Gândire algoritmică

- *În majoritatea situațiilor necesită gândire logică și* ***liberă***
- Să rezolvăm ***empiric*** următoarele probleme:
  - *Să se scrie un algoritm care afisează toate numerele naturale de la 1 la 100, inlcusiv.*
  - *Să se scrie un algoritm care însumează toate numerele naturale de la 1 la 100, inlcusiv.*
  - *Să se scrie un algoritm care însumează fiecare număr par de la 1 la 100, inlcusiv.*
  - *Să se scrie un algoritm care ordonează trei numere date.*

4. Algoritmi → programare

Algoritm adunare:

1. Se citește primul număr de pe tastatură
2. Se citește al doilea număr de pe tastatură
3. Se efectuază operația de adunare
4. Se salvează rezultatul într-un număr rezultat
5. Se afișează pe ecran numărul rezultat

Pseudocod

```
Algoritm adunare este
    Intreg a,b,c;
    Citeste a;
    Citeste b;
    c:=a+b;
    Afișează c;
Sfarsit algoritm
```

Limbaj programare

```
#include<stdio.h>
int main(){
    int a,b,c;
    scanf("%d",&a);
    scanf("%d",&b);
    c = a + b;
    printf("%d",c);
    return 0;
}
```

5. Probleme insolvabile algoritmic

- **Există probleme care nu pot fi rezolvate cu ajutorul calculatorului?**
- Răspunsul este DA:
  1. **Problema Stopului. Nu exista un algoritm universal valabil** prin care sa se poata decide daca executia oricarui algoritm se opreste vreodata sau nu.
  2. **Problema ecuatiilor diofantice. Nu exista o metoda generala (un algoritm)** de aflare a solutiilor întregi ale unui sistem de ecuatii diofantice. $p^4+q^4+r^4=s^4$
  3. **Problema acoperirii planului**
  4. **Problema sirurilor lui Post**
  5. **Problema cuvintelor "egale"**
     - **Ex:** EAT=AT, ATE=A, LATER=LOW, PAN=PILLOW si CARP=ME.

       Este CATERPILLAR egal cu MAN ?

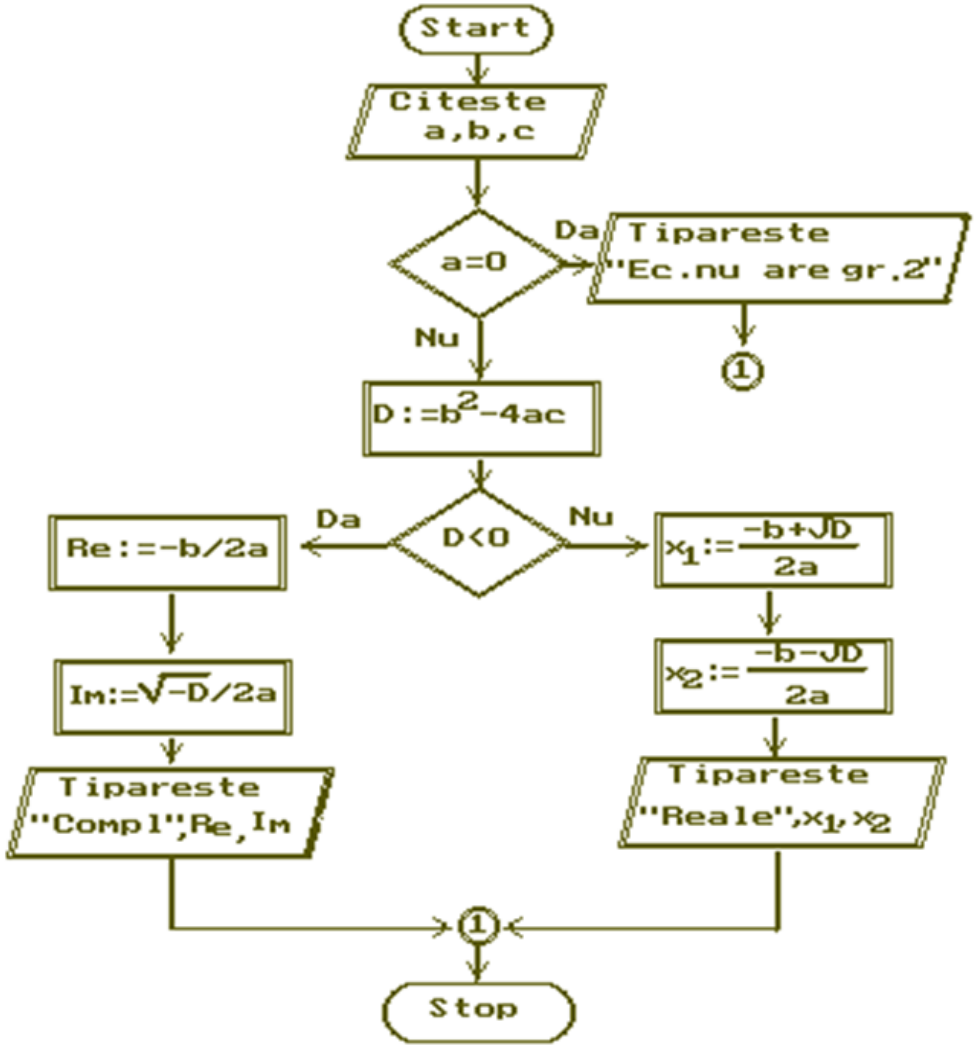
CATERPILLAR = C***A***RPILLAR =CARPILL***ATE***R=CARPIL***LOW***= CAR***PAN***= ***ME***AN= ME***ATE***N= M***AT***EN= M***A***N

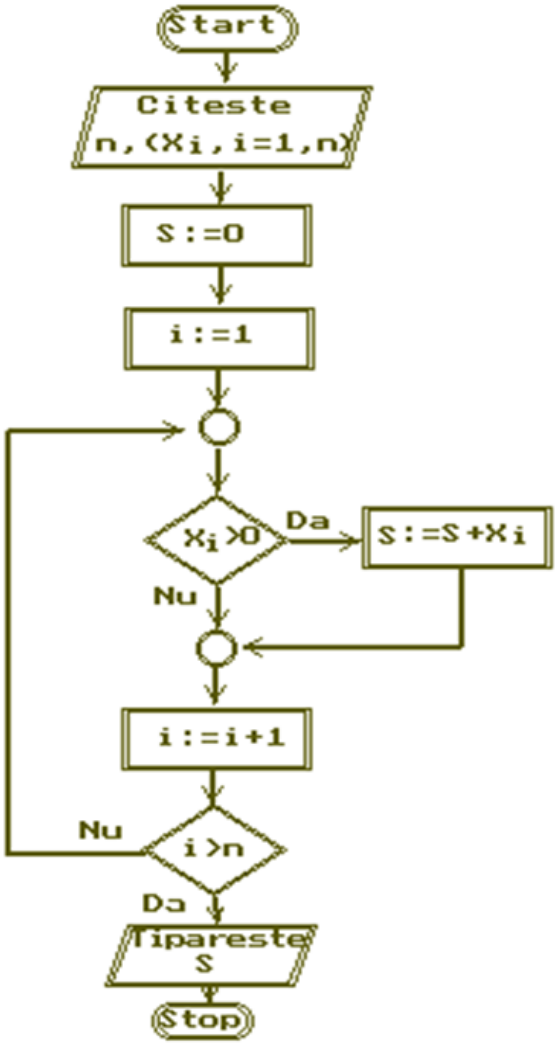
AH=HA, OH=HO, AT=TA, OT=TO, TAI=IT, HOI=IH si THAT=ITHT?

6. Descrierea algoritmilor

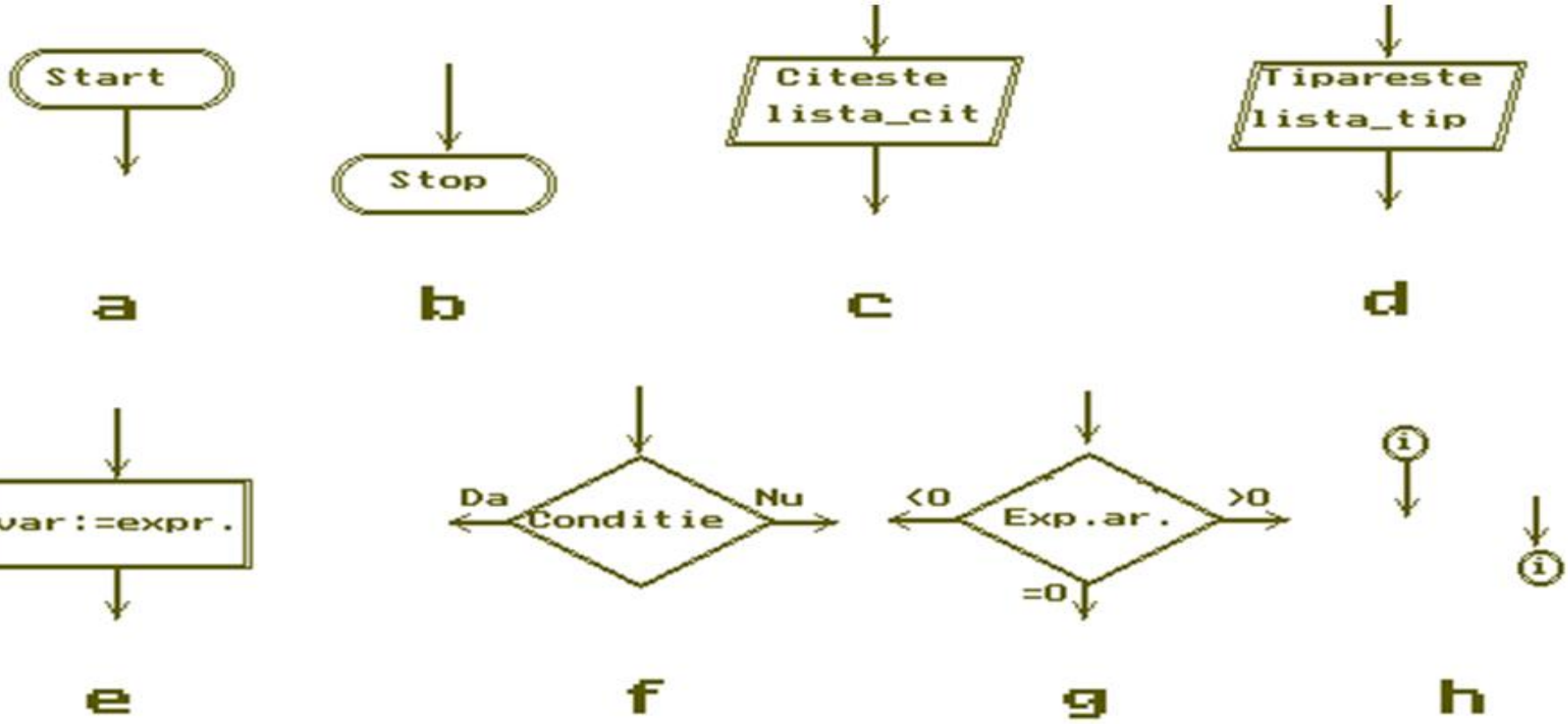
- **Prin scheme logice (desen)**
  - Descrie algoritmul pe baza unui set bine definit de simboluri grafice
  - Ușor de urmărit și de transcris într-un limbaj de programare
- **Prin Pseudocod (text)**
  - Descrie algoritmul pe baza unui set bine definit de cuvinte cheie și instrucțiuni

6. Scheme logice

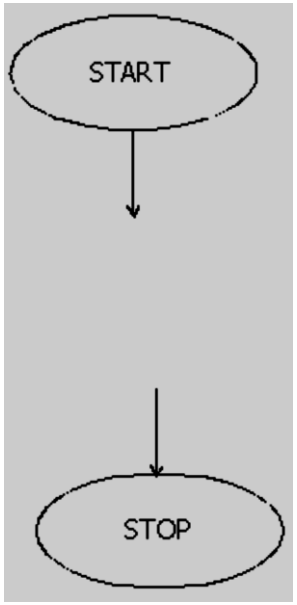




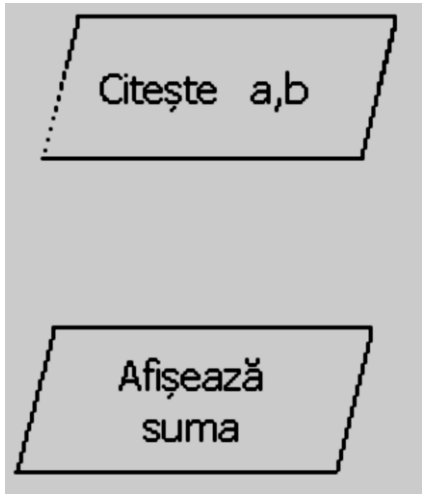
6. Scheme logice



6. Scheme logice

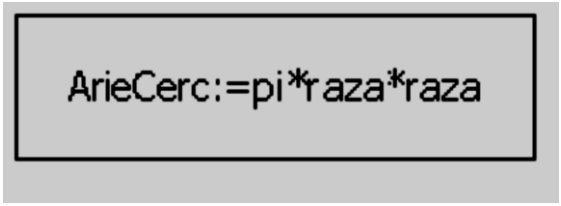


- Blocuri terminale



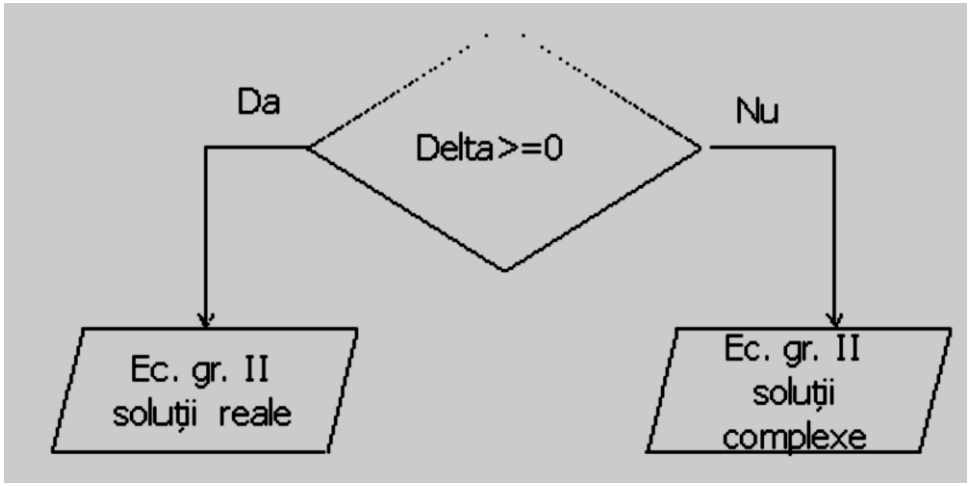
- Blocuri de intrare și ieșire

- Blocuri de calcul (atribuire)



6. Scheme logice

- Blocuri de decizie (ramificație, condiție)



- Blocuri de prelucrare complexă
  - subscheme

7. Probleme

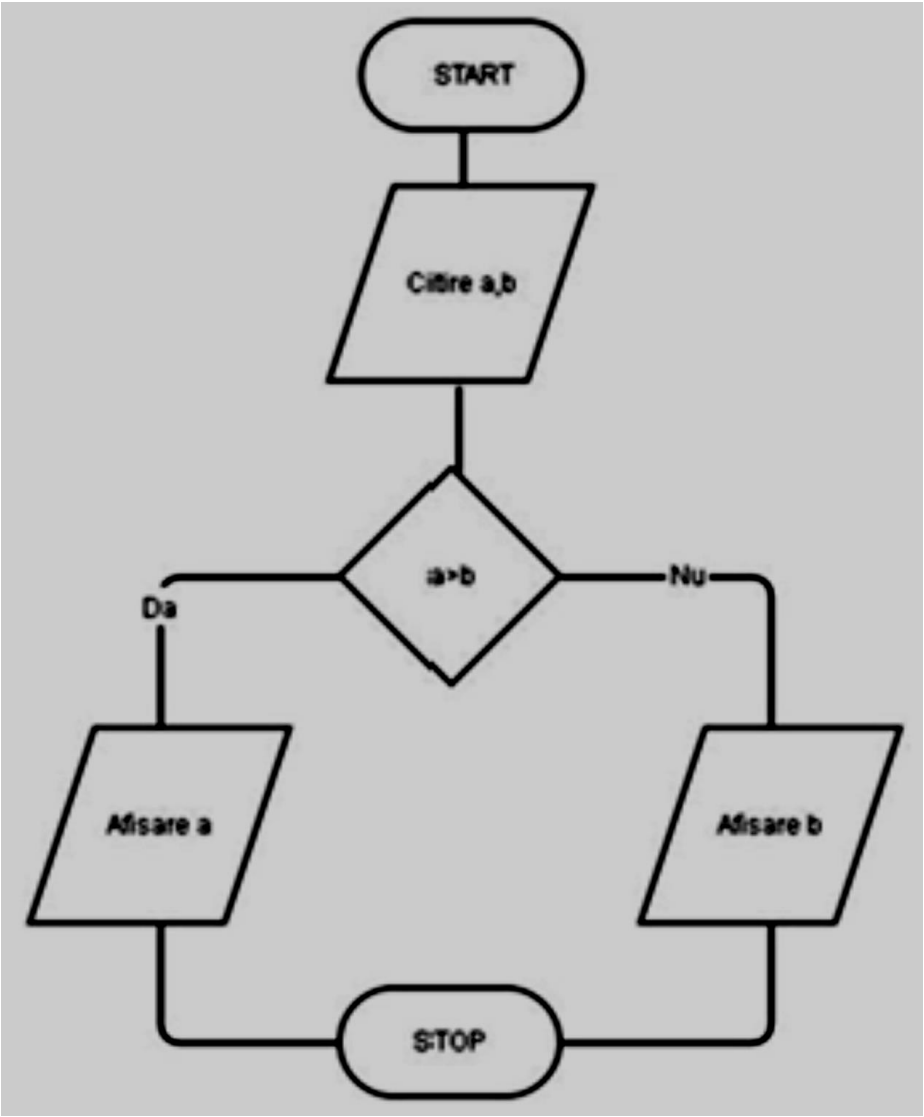
1. Algoritm de postare poză pe social media
2. Algoritm de schimbare poză de profil
3. Algoritm de reîncărcare credit telefon

- Ce se observa daca incepem sa rezolvam prin cate un algoritm problemele de mai sus???
  - Ce operatii elementare folosim?
  - Putem da o rezolvare formata dintr-un set de pasi finiti?

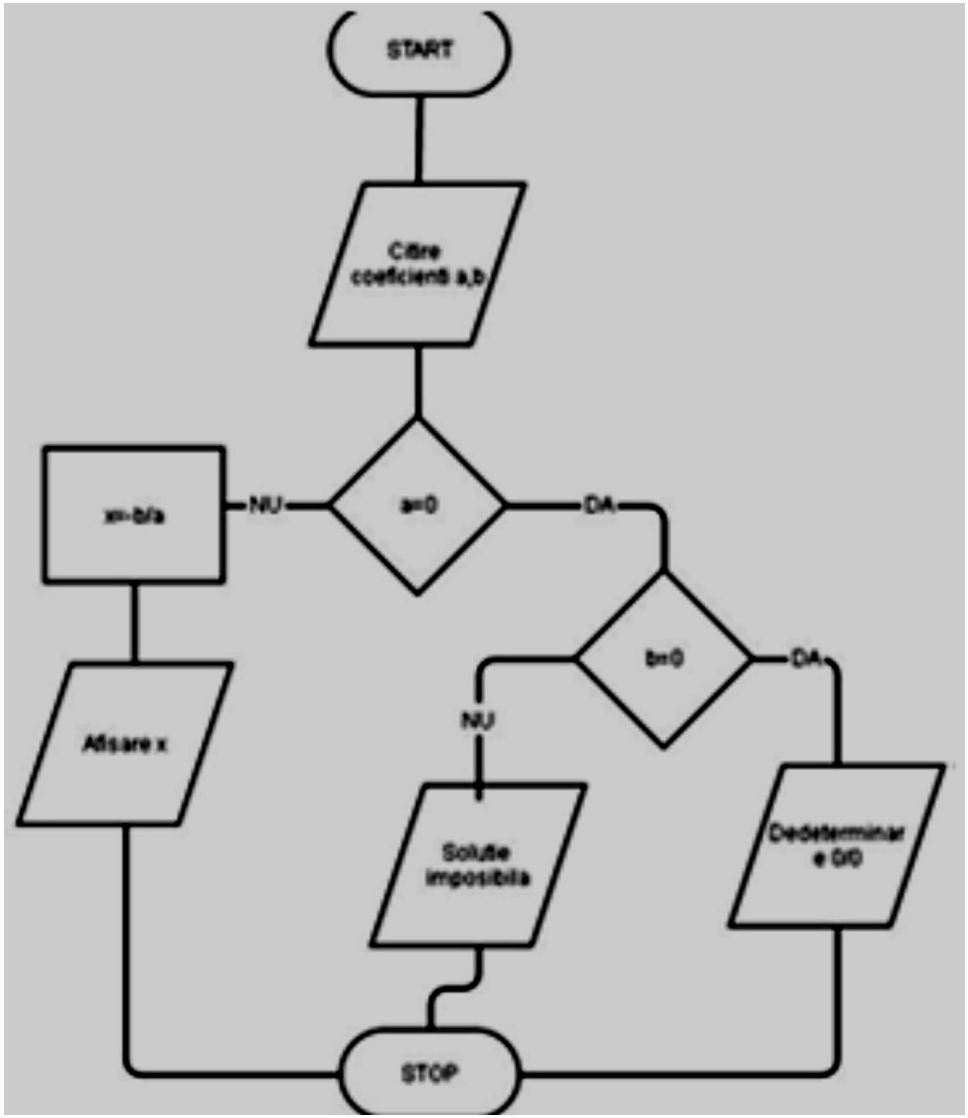
Probleme

1. Maximul a două numere naturale
2. Soluția ecuației de gradul I
3. Suma numerelor naturale de la 1 la 10
4. Afișarea factorialului unui număr dat
5. Soluția ecuației de gradul II
6. Să se calculeze suma elementelor pozitive ale unui şir de numere reale dat
7. Algoritm pentru operația ȘI binar

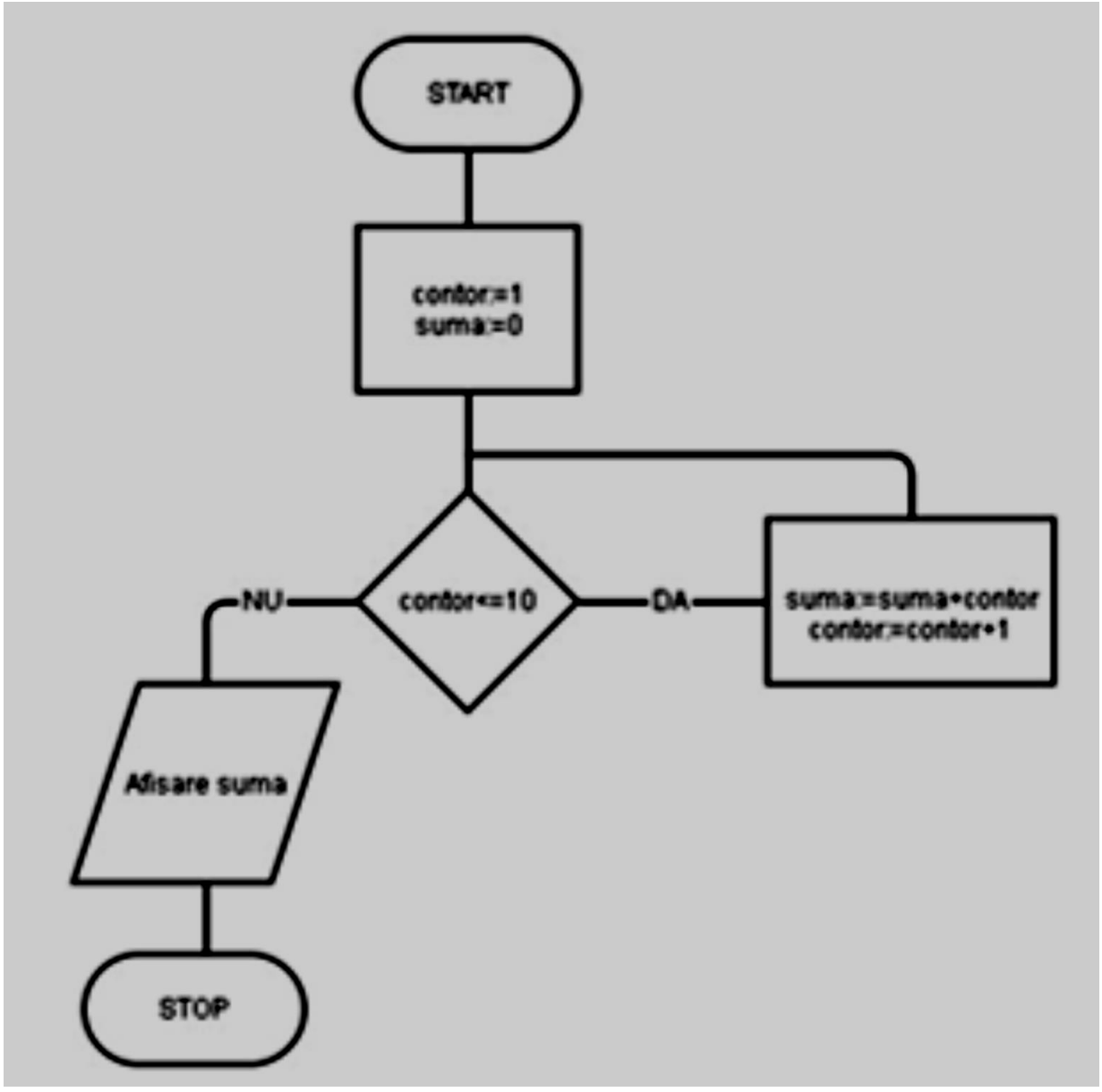
Maximul a două numere



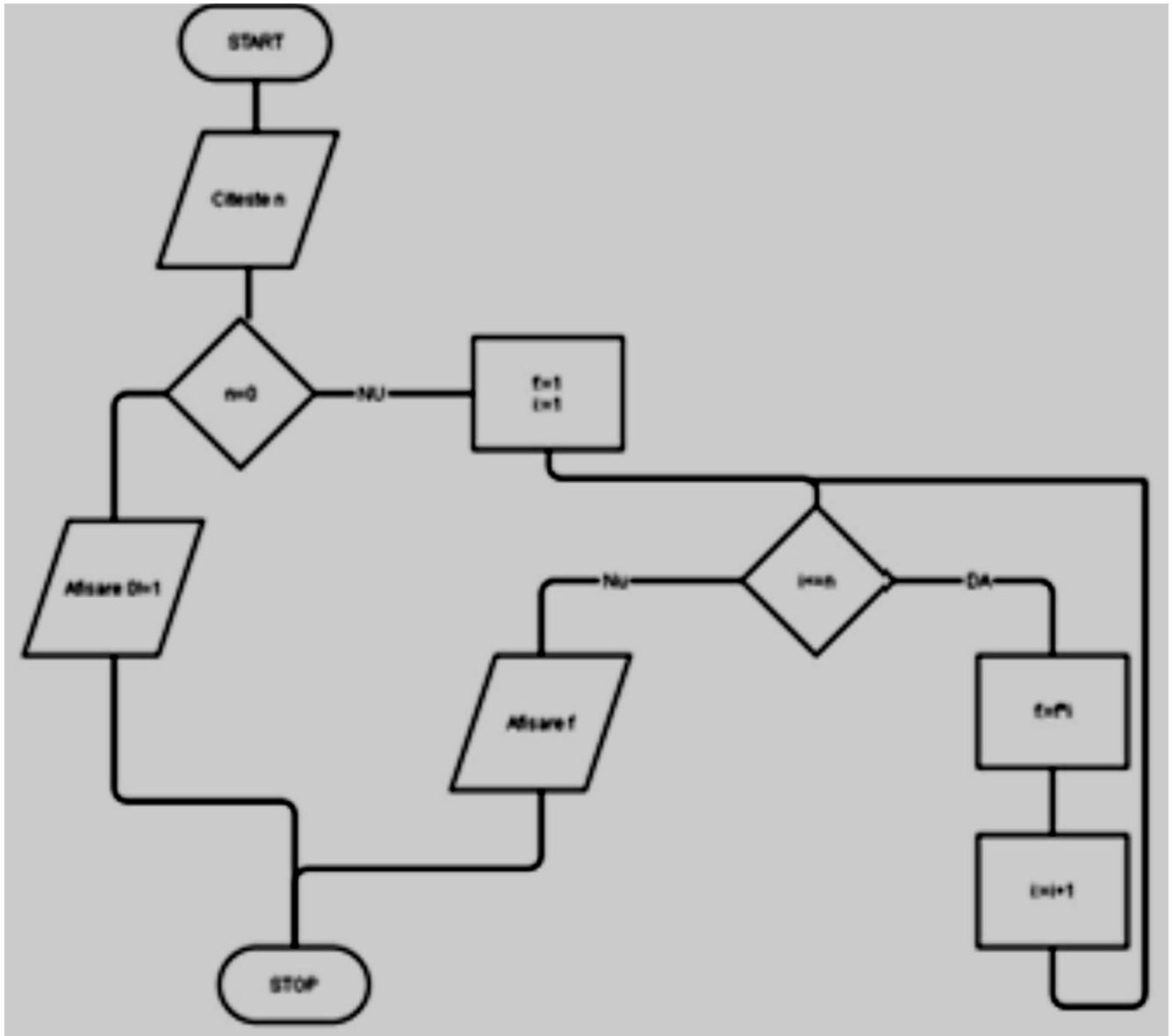
Soluția ecuației de gradul I



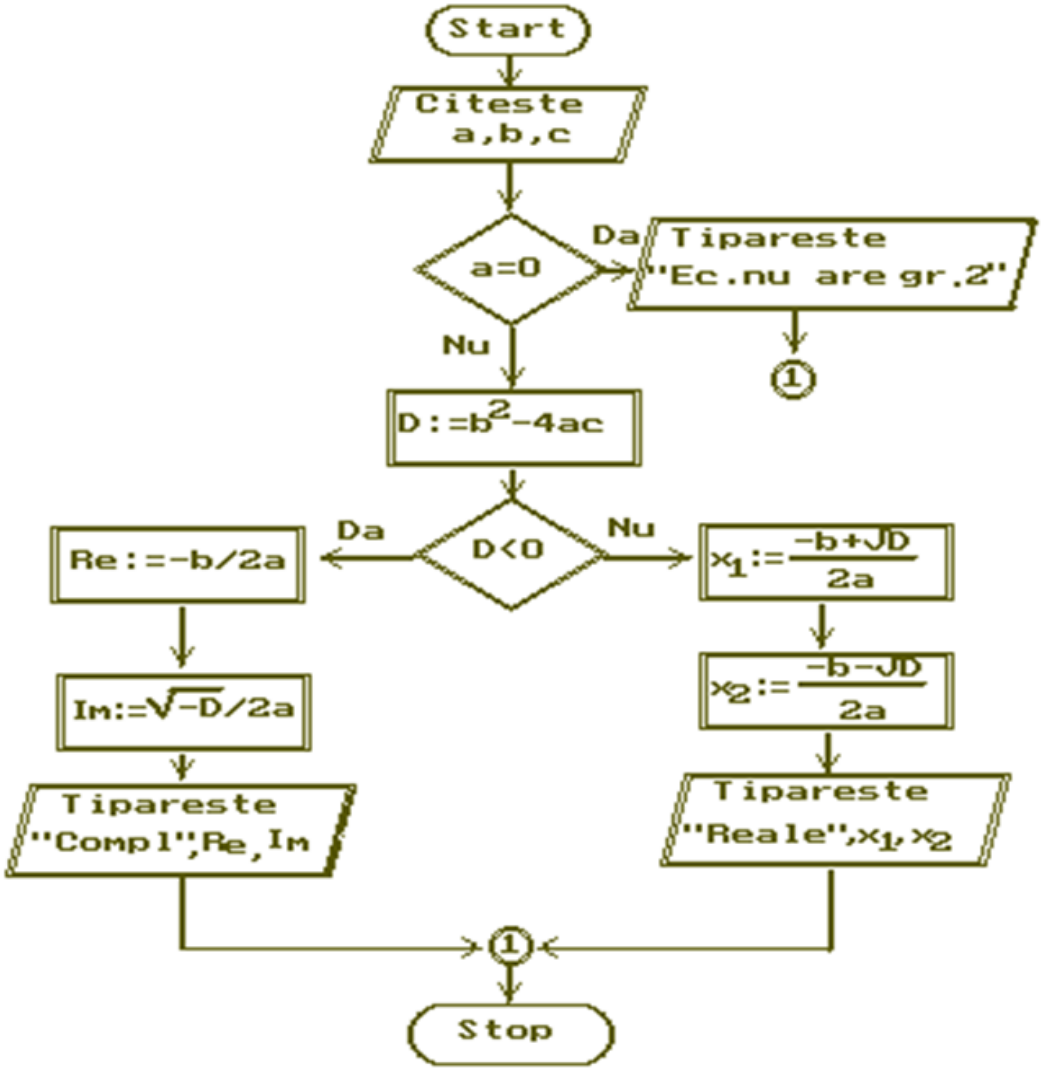
Suma numerelor naturale de la 1 la 10



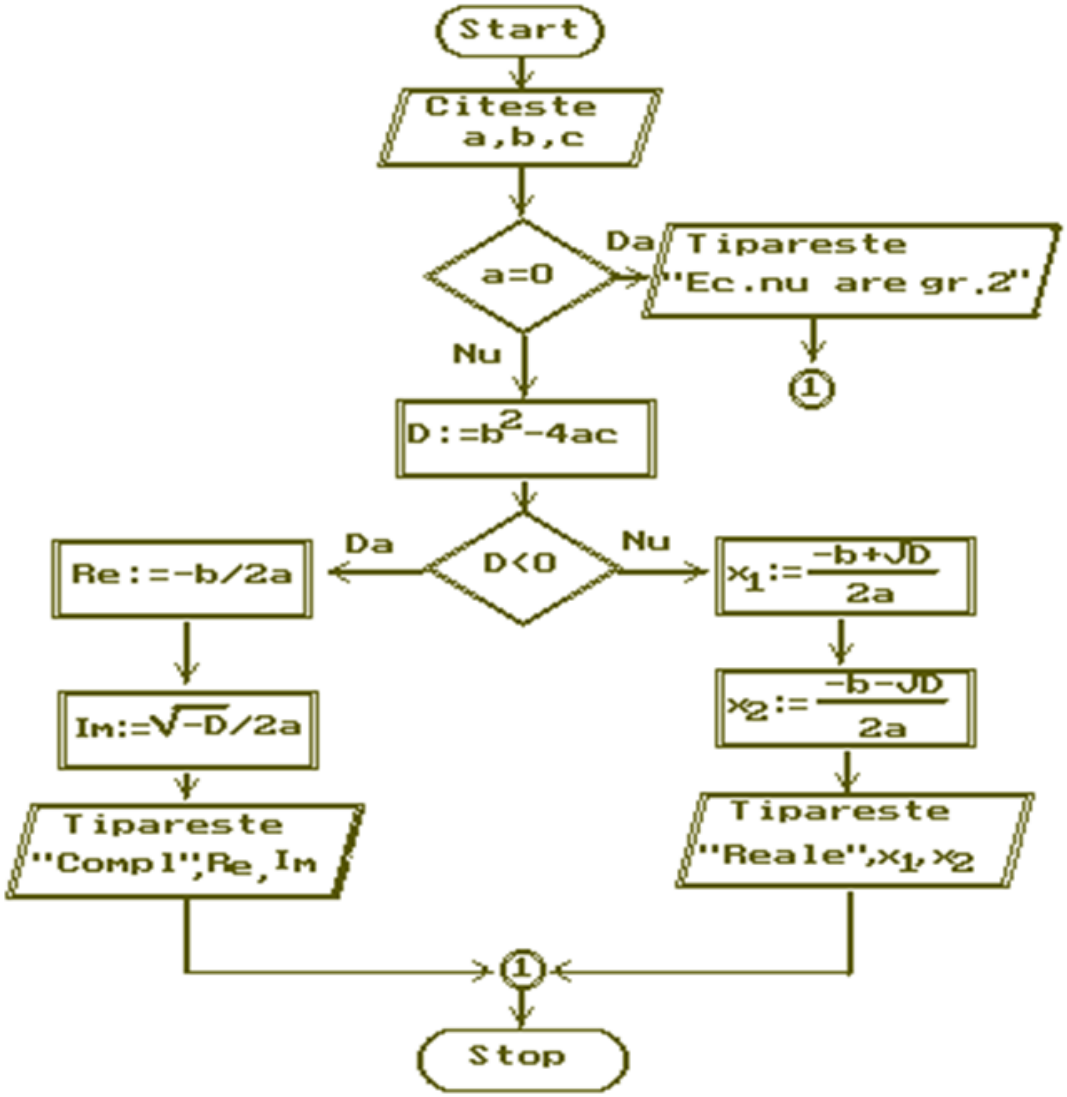
Afișarea factorialului unui număr dat



Soluția ecuației de gradul II



Să se calculeze suma elementelor pozitive ale unui şir de numere reale dat



Algoritm pentru operația ȘI binar