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

Curs 1

Dr. ing. Kiss István

istvan.kiss@umfst.ro

Curs + seminar/laborator

- 2 ore
  - Se prezintă teoria de bază și se explică prin probleme practice

- 2 ore seminar/laborator / 3 grupe
  - Se rezolvă probleme împreună cu studenții
  - Se discută eventualele aspecte de rezolvare
  - Se transpun algoritmii elementari studiați în limbaje de programare
    - Skratch, AppInventor, C (Online sau Codeblocks)

Nota finală

- 50% examen din materia predată la curs
- 50% nota de laborator
- 10% prezență și activitate la curs/laborator

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

Bibliografie

- [1] Enăchescu, C. – *Structuri de date și algoritmi*. Editura Casa Cărții de Știință, 2004.
- [2] Stephens, R. – *Essential algorithms*. Editura John Wiley & Sons, 2013.
- [3] Sedgewick, R. – *Algorithms in C.* Editura Addison-Wesley, Massachusetts, 1990.
- [4] Lewis, H.R., Denenberg, L. – *Data structures & their algotihms*. Editura Harper Collins Publishers, 1991.
- [5] Andonie, R., Gârbacea, I. – *Algoritmi fundamentali. O perspectivă C++.* Editura Libris, Cluj-Napoca, 1995.
- [6] Erickson, J. – *Algorithms*. Lecture notes, Illinois, 2009.
- [7] Brassard, G., Bratley, P. *Algorithmics - Theory and Practice*. Prentice-Hall, Englewood Cliffs, 1988.
- [8] Baase, S. – *Computer algorithms. Introduction to design and analysis*. Editura Addison-Wesley, 1987.
- [9] *** - *A didactic, animated, exposition of algorithms*. Disponibil Online: www.algomation.com

Conținut curs

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 curs – fisa disciplinei

- **Introducere în algoritmi. Definiții. Gândire algoritmică. Legătură între algoritmi și programare. Probleme solvabile algoritmic. Descrierea algoritmilor. Motivarea studiului de algoritmi prin exemple și studii de caz.**
- **Pseudocod. Bazele logicii binare. Operații de bază. Propoziții binare. Date intrare-ieșire. Noțiunea de variabilă. Algoritmi liniari. Algoritmi cu ramificații. Algoritmi ciclici. Algoritmi elementari folosind cele trei tipuri de algoritmi. Probleme.**
- **Tehnici pentru elaborarea algoritmilor. Subalgoritmi. Apel de subalgoritm. Avantajele subalgoritmilor. Proiectare ascendentă (top-down) și descendentă (bottom-up). Proiectare modulară. Probleme.**
- **Analiza algoritmilor. Etapele rezolvării problemelor. Exemplificare prin probleme. Funcții de complexitate. Clase de complexitate. Determinarea complexității unui algoritm dat.**
- **Recursivitate. Algoritmul recursiv. Paradigma Divide-et-Impera. Exemplificarea recusivității prin turnurile din Hanoi. Comparație algoritm recursiv și nerecursiv. Probleme.**
- **Algoritmi de sortare. Tablouri unidimensionale și bidimensionale. Algoritmi elementari (prin selecție, metoda bulelor, prin inserție, sortare shell). Algoritm de sortare rapidă. Sortare radix. Sortare prin interclasare și prin ansambluri (heapsort). Probleme.**
- **Analiza algoritmilor de sortare. Recapitularea algoritmilor studiați. Determinarea complexității. Clasarea algoritmilor în funcție de complexitatea lor. Discuții și analiza algoritmului de sortare cu numărarea frecvențelor (counting sort) și a algoritmului de sortare cu găleți (Bucket sort).**
- **Algoritmi de căutare. Căutare secvențială. Căutare binară. Căutare prin interpolare. Ștergere element din tablou. Complexitatea algoritmilor studiați.**
- **Operații pe șiruri de caractere. Referire la un element. Tablou (Vector) de șiruri de caractere. Subprograme de prelucrare (lungime, copiere, concatenare, comparare, inserare, căutare, ștergere, tokenizare). Subprograme de conversie (din șir în număr și invers). Probleme.**
- **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.**

Cuprins primul curs

1. Introducere în algoritmi
2. Definiții
3. Gândire algoritmică
4. Algoritmi → programare
5. Probleme insolvabile algoritmic
6. Descrierea algoritmilor
7. Probleme

1. Introducere

- Trăiți zilele după un algoritm sau nu?
  - Vă treziți dimineața
  - Mergeți la facultate
  - După masa vă întâlniți cu prietenii
  - Seara poate aveți o întâlnire cu prietena/prieten
  - Vă puneți în pat
  - ... Și se începe a doua zi
- Sau aveți o viață fără ”program” (algoritm)?
- Spre deosebire de noi, calculatorul știe **numai** să execute programe predefinite alcătuite din secvențe finite de instrucțiuni

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

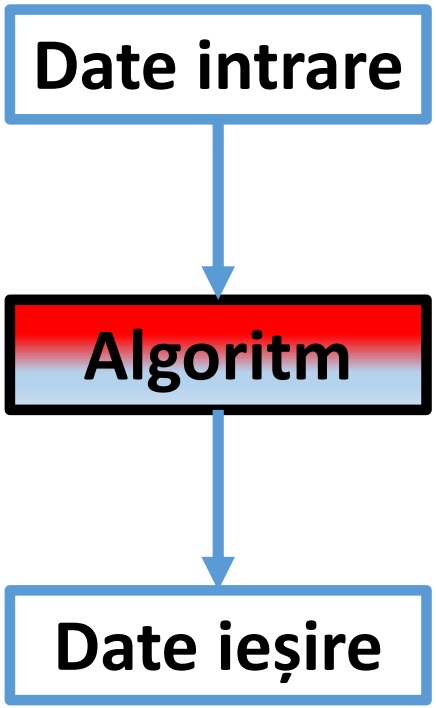
    Intreg a,b,c; - *date intrare și ieșire*

    Citeste a;

    Citeste b;

    c:=a+b;

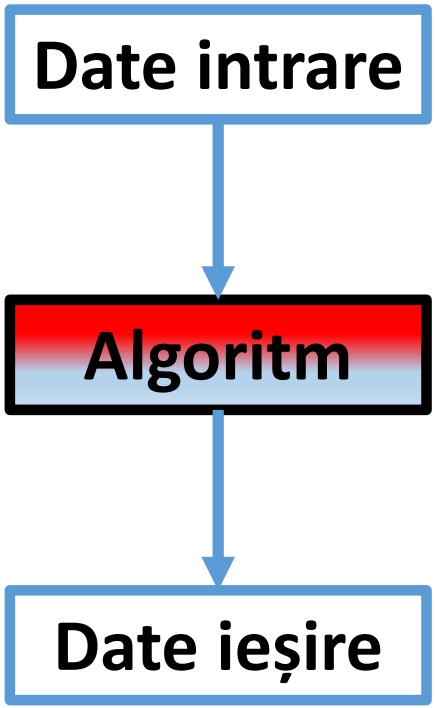
    Afișează c;
  - Sfarsit algoritm



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;
}
```



2. Definiția algoritmului

- Algoritmul este un set finit de pași executabili (operații elementare), descriși fără echivoc, care prelucrează datele de intrare în scopul soluționării unei clase de probleme

- Este metoda de rezolvare a unei probleme
- Este independent de limbajul de programare
- Orice algoritm pornește de la date de intrare și dorește obținerea unui rezultat

2. Proprietățile algoritmilor

- **Finitudine**: un număr finit de pași
- **Claritate**: algoritmul trebuie descris clar, fără ambiguități
- **Generalitate**: trebuie să rezolve toate problemele dintr-o clasă de probleme
- **Determinare**: fiecare pas care urmează poate fi determinat în mod unic pe baza pașilor sau rezultatelor precedente
- **Eficiența**: timpul de execuție și spațiul de memorie utilizat
- **Executabil**: fiecare pas poate fi codificat într-un limbaj de programare

Exemplu de algoritm gresit

- Cum sa ajungem de la etajul 2 la etajul 5 cu liftul?
  1. `Apasa butonul de chemare!`
  2. `Urca-te in lift!`
  3. `Apasa butonul 5!`
  4. `Asteapta!`
  5. `Daca usa se deschide, coboara-te!`

- Probleme (**finititudine**):
  - Ce daca intamplator liftul chemat coboara?
  - Ce daca liftul se opreste la etajul 3 pentru a se urca altcineva?

Exemplu de algoritm gresit

- Cum sa fim milionari?
  1. `Cumpara un lotto!`
  2. `Pune numerele dorite!`
  3. `Asteapta pentru castig (ori fi nervos)!`

- Probleme (aleatoriu, nu e deterministic):
  - In majoritatea cazurilor nu furnizeaza rezolvare.
  - Numai uneori este valibil ca si o rezolvare a problemei.

Exemplu de algoritm gresit

- Cum sa circulam cu autocarul?
  1. `Asteapta!`
  2. `Urca-te!`
  3. `Cumpara-ti bilet!`
  4. `Stai jos!`
  5. `Daca ai ajuns, coboara-te!`

- Probleme (claritate, finititudine):
  - Daca nu esti intr-o statie, autocarul nu va opri.
  - Daca te urci pe un autocar gresit, nu mai poti cobora deloc (conform instructiunilor din algoritm).

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 ?

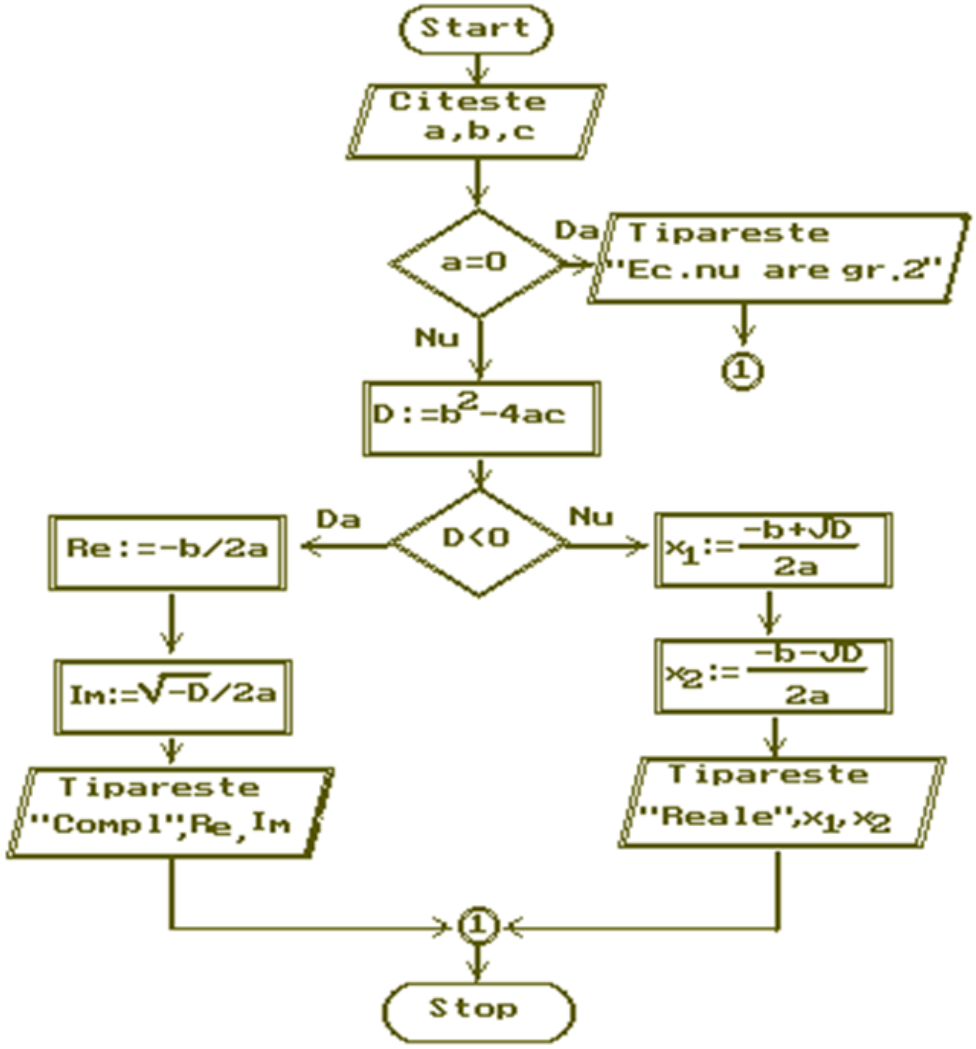
CATERPILLAR = C***A***RPILLAR =CARPILL***ATE***R=CARPIL***LOW***= CAR***PAN***= ***ME***AN= ME***ATE***N= M***AT***EN= M***A***N

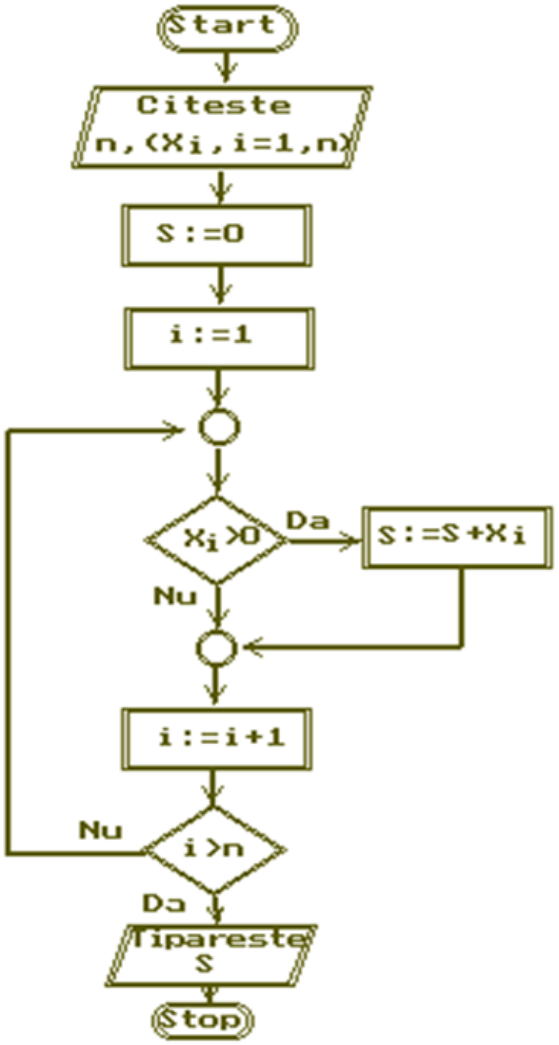
AH=HA, OH=HO, AT=TA, OT=TO, TAI=IT, HOI=IH si THAT=ITHT?

6. Descrierea algoritmilor

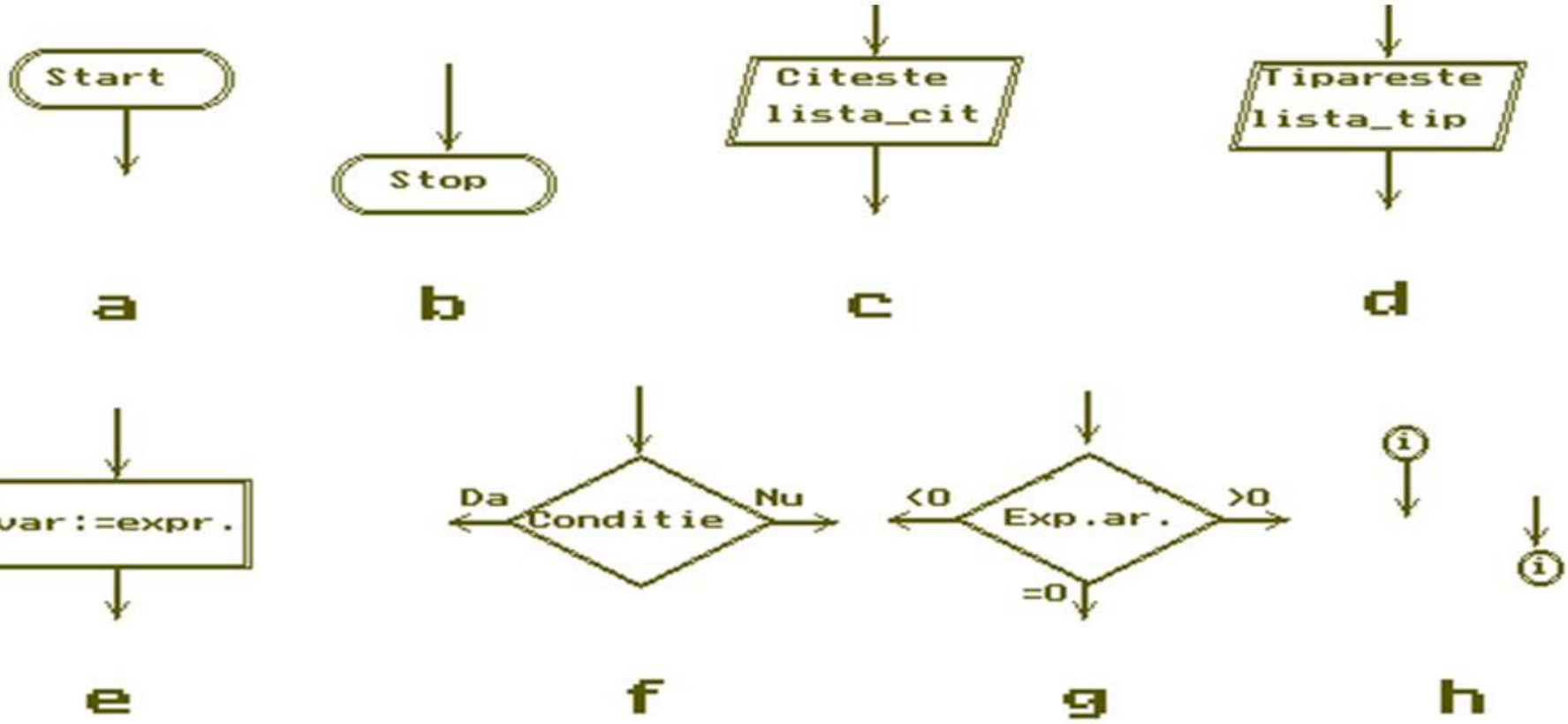
- Prin scheme logice (desen)
  - Descrie algoritmul pe baza unui set bine definit de simboluri grafice
  - Ușor de urmărit și de transcris într-un limbaj de programare
- Prin Pseudocod (text)
  - Descrie algoritmul pe baza unui set bine definit de cuvinte cheie și instrucțiuni

6. Scheme logice

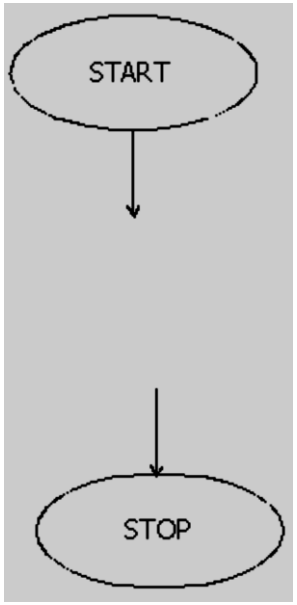




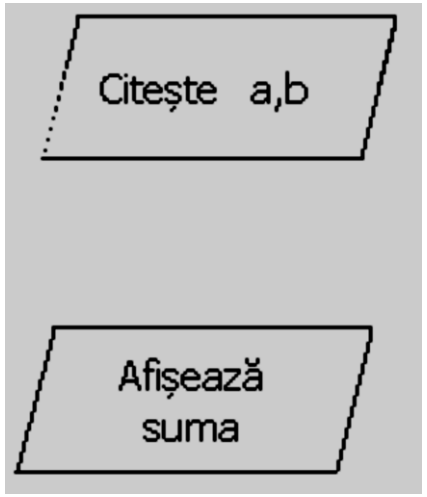
6. Scheme logice



6. Scheme logice

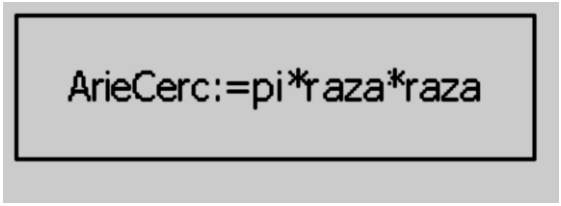


- Blocuri terminale



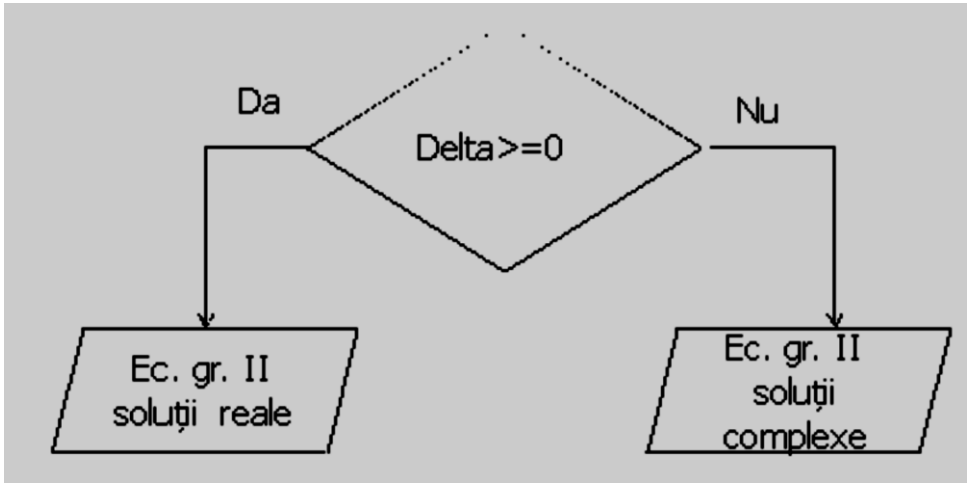
- Blocuri de intrare și ieșire

- Blocuri de calcul (atribuire)



6. Scheme logice

- Blocuri de decizie (ramificație, condiție)

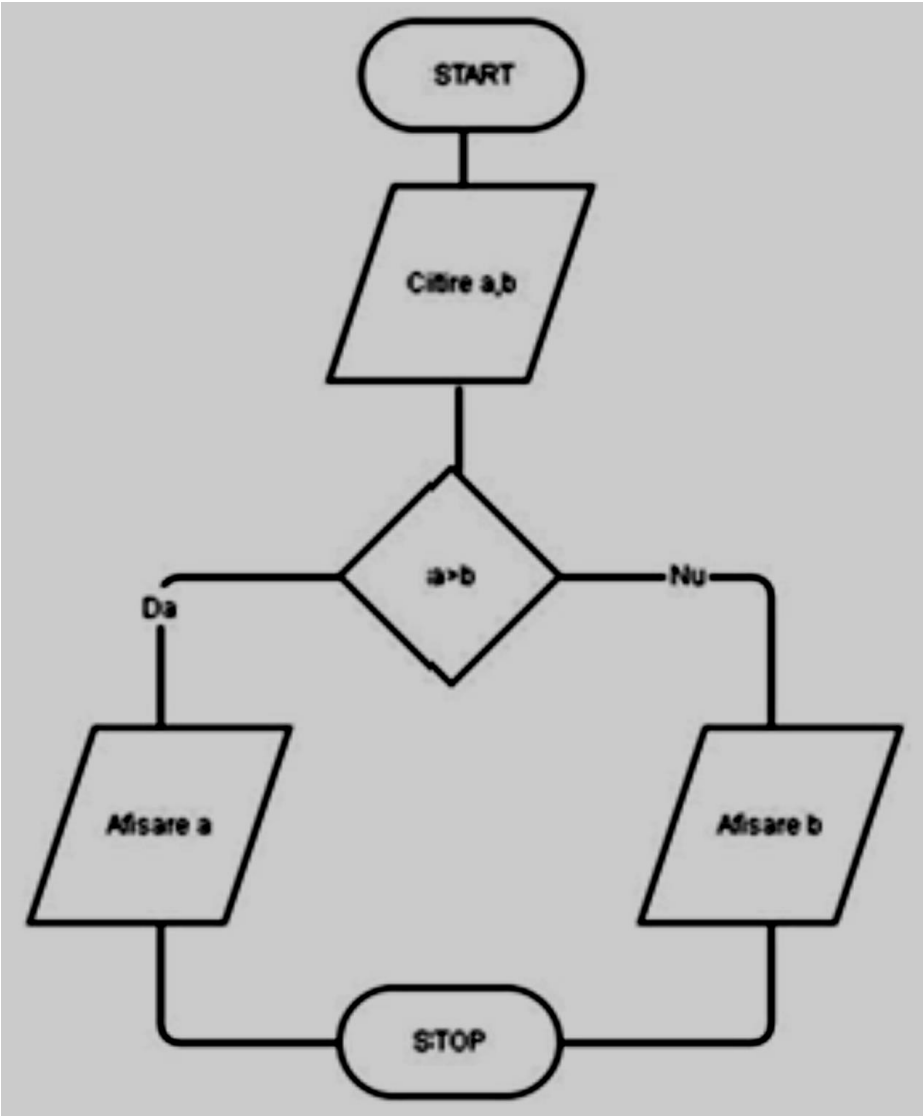


- Blocuri de prelucrare complexă
  - subscheme

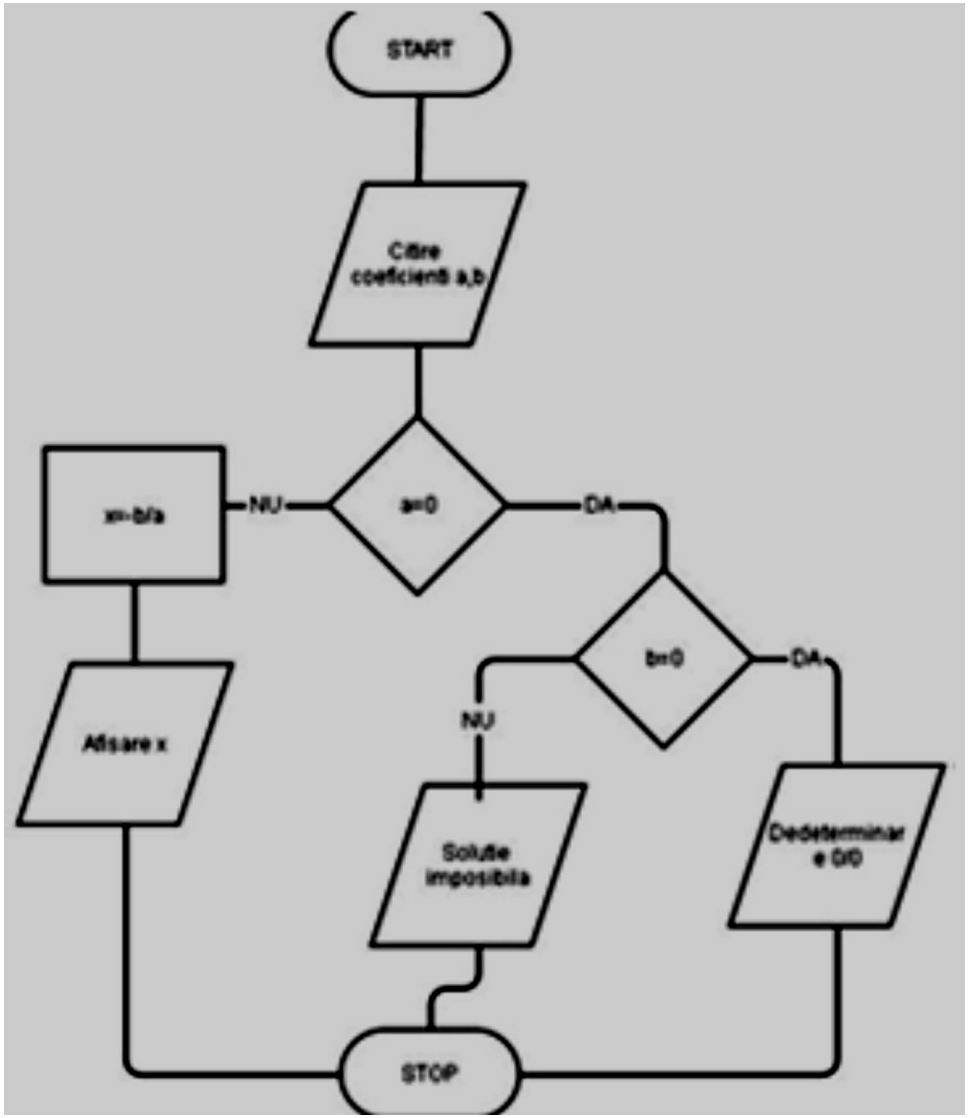
7. Probleme

1. 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

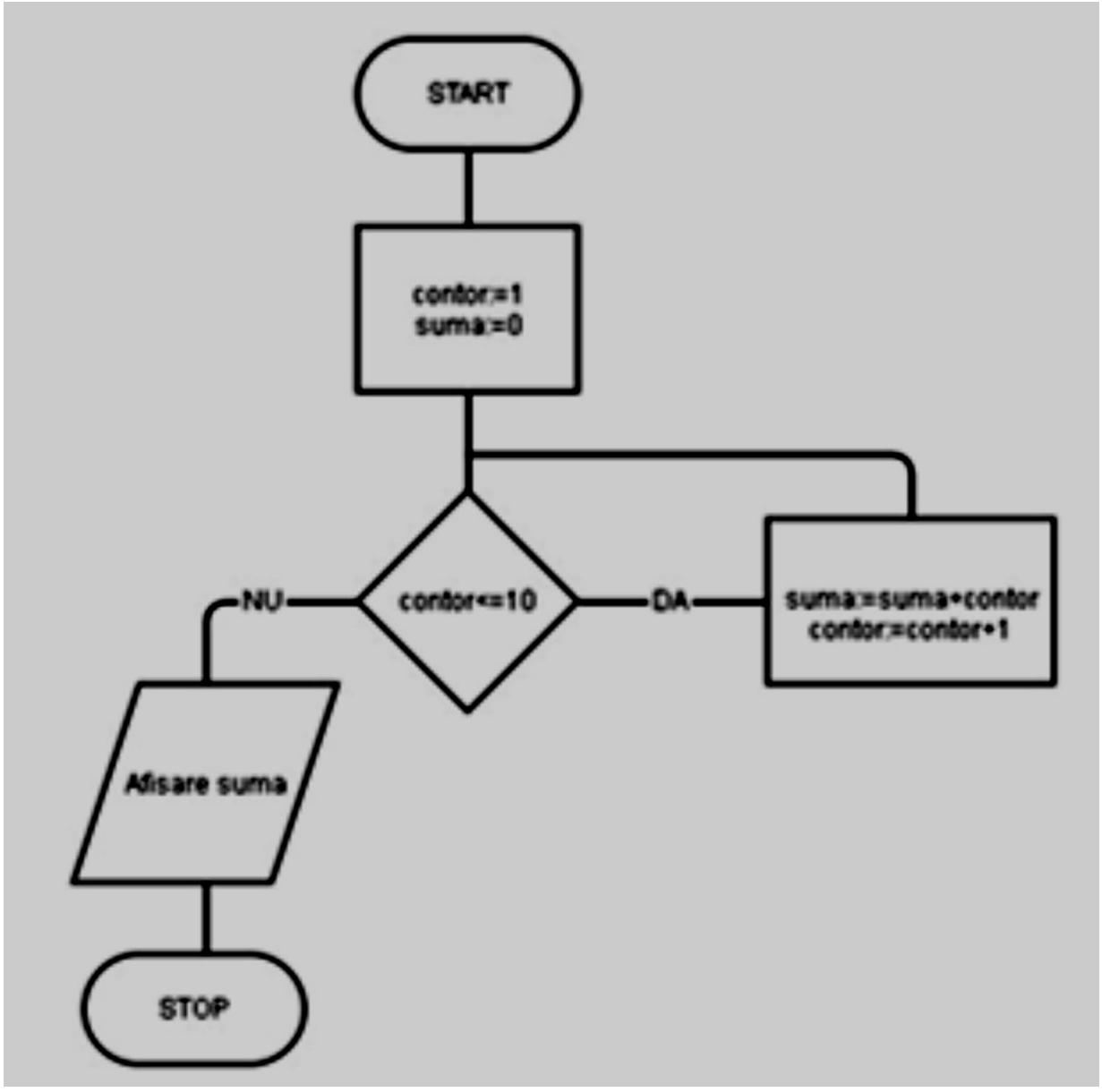
Maximul a două numere



Soluția ecuației de gradul I



Suma numerelor naturale de la 1 la 10



Afișarea factorialului unui număr dat