

# Structuri de date și algoritmi

---

DR. ING. CRISTINA STÂNGACIU

# Introducere

---

## CAPITOLUL I

# Cuprins

---

Obiectivele cursului

Despre curs

Introducere

Noțiunea de structuri de date

Noțiunea de algoritm

Scheme logice

Exerciții

Bibliografie selectivă

# Obiectivele cursului

---

## Obiective principale:

- Prezentarea structurilor de date ca **tipuri de date abstracte** în strânsă interdependență cu **algoritmii** care implementează operațiile specifice definite pe aceste tipuri
- **Analiza eficienței** structurilor de date și a algorimilor

## Obiective secundare:

- Deprinderea abilităților de:
  - analiză
  - proiectare
  - implementare
- a structurilor de date în diverse contexte aplicative
- a unor algoritmi, utilizând operațiile de bază ale structurilor de date studiate

# Despre curs

---

## Ce vom găsi în acest curs?

- Prezentarea structurilor de date fundamentale ca tipuri de date abstracte
- O introducere în domeniul algoritmilor
- Algoritmi ce implementează operații specifice structurilor de date fundamentale
- Metode de analiză, proiectare și implementare a algoritmilor folosind structuri de date fundamentale
- Exemple de implementari ale unor algoritmi de referință

# Despre curs

---

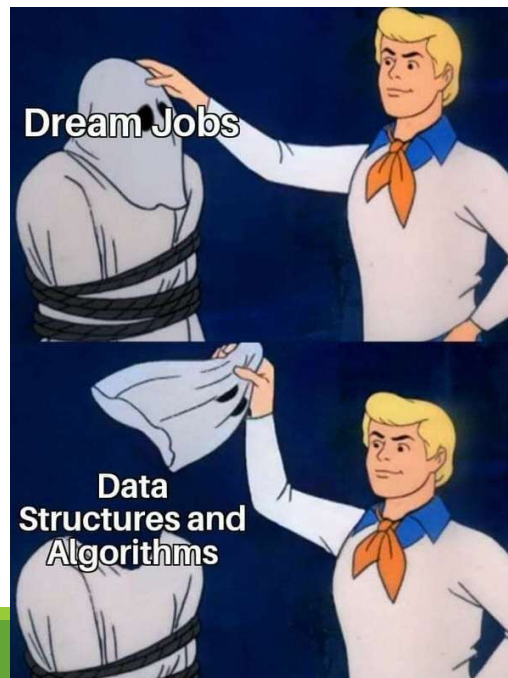
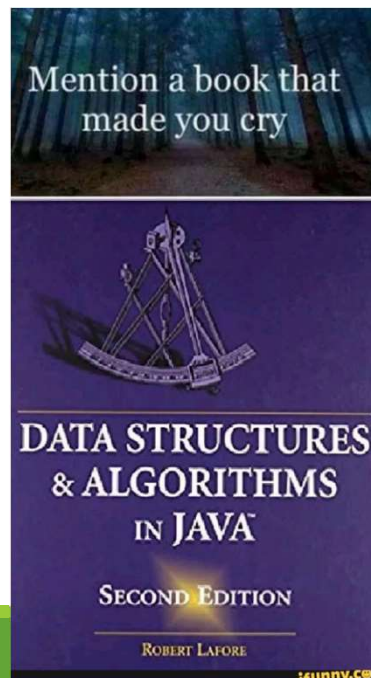
## De ce să studiem SDA?

- La materiile anterioare de programare am învățat principii de programare, la SDA continuăm studiul acestor principii, adăugând eficiența ca obiectiv principal
- SDA este o materie de bază pentru alte materii ca Proiectarea și analiza algoritmilor; Programarea orientată pe obiecte și diverse materii opționale din ani mai mari
- Este o materie de bază la toate universitățile mari ale lumii pentru profilul Calculatoare (Computer Science) și Tehnologia Informației (Information Technology)
- Locuri de muncă bine plătite în domeniul dezvoltării de algoritmi de securitate, de telecomunicații, inteligență artificială etc.
- Pentru a ne dezvolta gândirea algoritmică

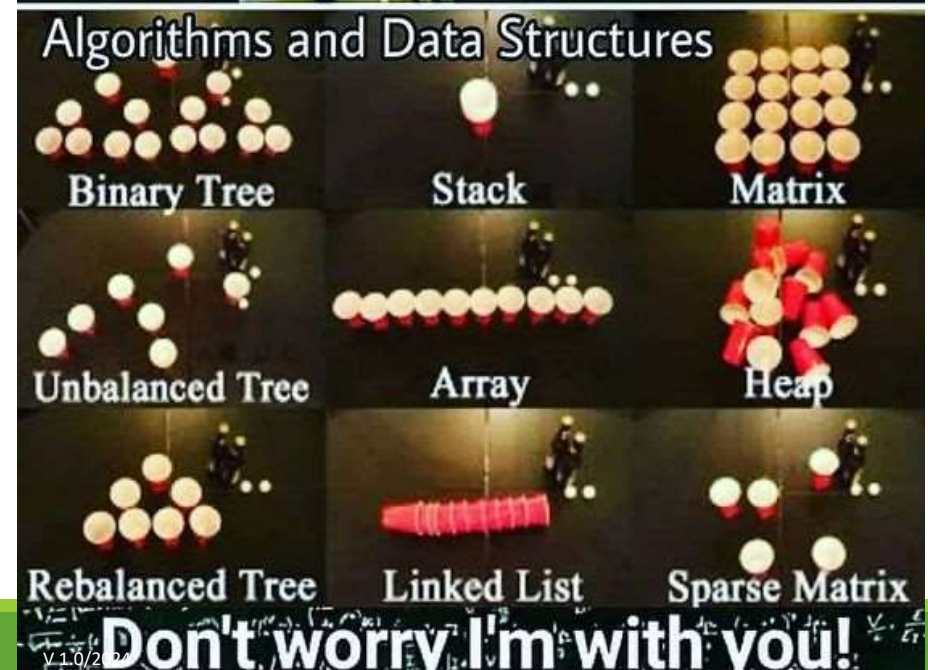
# Despre curs

De ce să studiem SDA?

- Și ... pentru a înțelege glumele cu structuri de date și algoritmi de pe rețelele de socializare ☺



She: I'm sad, nobody understand me!



# Despre curs

---

## **Cum se va desfășura acest curs?**

- Sesiuni de curs – 14 \* 2 ore
- Sesiuni de laborator - 14 \* 2 ore
- Consultații la cerere, după un orar stabilit în prealabil
- Notare:
  - Examen – în sesiune
  - Activitate pe parcurs – activități de laborator notate
- Mai multe detalii pe [cv.upt.ro](http://cv.upt.ro) în cadrul secțiunii dedicate cursului (v. Regulamentul cursului de SDA și Regulamentul laboratorului de SDA).



# Despre curs

---

## Condiții preliminare

- Acest curs se bazează pe cursurile de programare și de analiză matematică.
- Condițiile preliminare acestui curs sunt:
  - **o bună cunoaștere a limbajului de programare C și a tehnicilor de programare în general!**
  - **cunoștințe de bază** privind limite, serii, logaritmi, recursivitate și demonstrații matematice!

# Despre curs

---

## Principii de bază!

- Pentru a fi bun în domeniul algoritmilor (în general) și pentru acest curs (în mod particular) avem nevoie de:
  - **Atenție** – neatenția duce la rezultate eronate
  - **Întelegere** – conceptele și regulile trebuie înțelese, nu învățate mecanic
  - **Gândire** – rezolvarea problemelor presupune o gândire analitică și algoritmică
  - **Exercițiu** – modul de gândire algoritmică se dezvoltă prin mult exercițiu (laboratoare și teme de casă)

# Despre curs

---

## Un pic de etică!

Ce înseamnă *plagiat* în contextul acestei materii?

- Copierea soluțiilor problemelor din diferite surse (incluzând ChatGPT, Bard sau alte sisteme generative) sau prezentarea muncii altor persoane ca fiind una proprie este o formă de **plagiat** și va fi sancționată conform regulamentului universității!
- Ce efecte negative are plagiatul?

Ce înseamnă *integritate* în activitatea profesională?

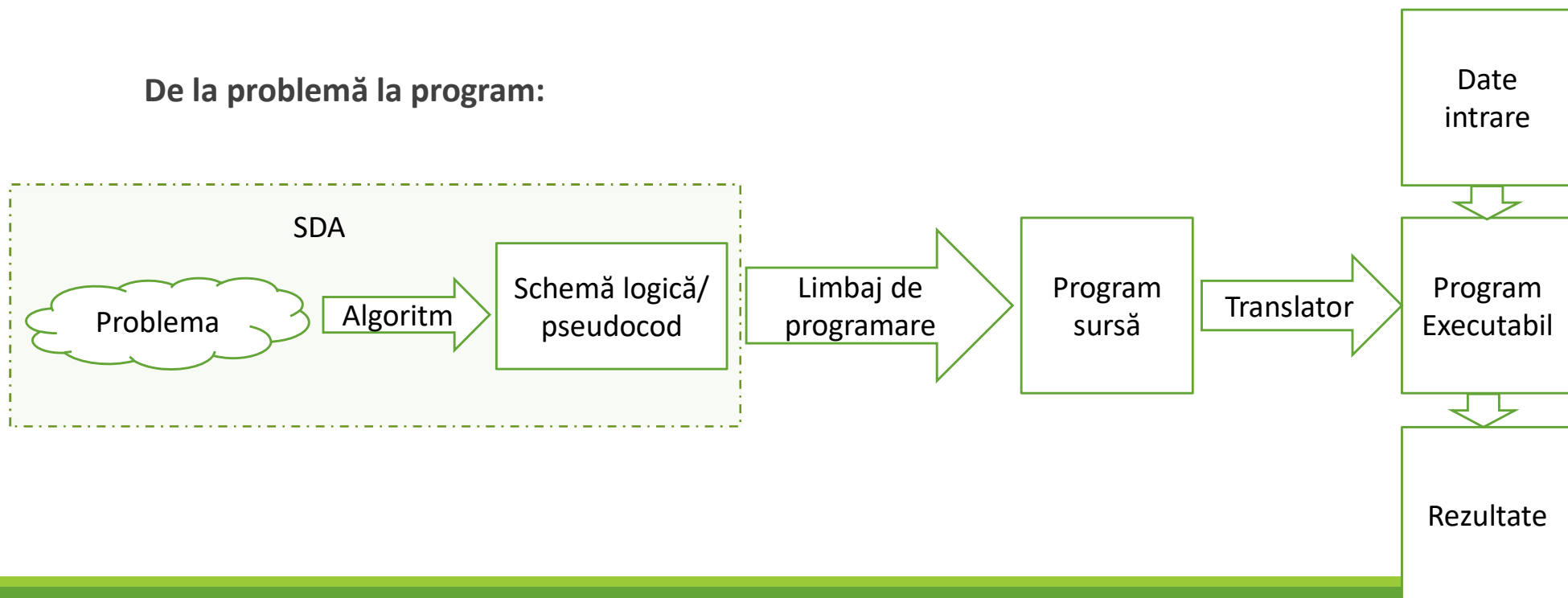
- "Integritatea este acțiunea comportament onorabilă, chiar și atunci când nimeni nu te privește."<sup>1</sup>

<sup>1</sup> [HTTPS://ASOCIATIAGETICA.RO/ETICA-PROFESIONALA/](https://asociatiagetica.ro/etica-profesionala/)

# Despre curs

---

**De la problemă la program:**



# Introducere

---

Câte orașe din țară au mai mult de 100 000 de locuitori? Cât cablu necesită conectarea la o rețea a tuturor orașelor din județ?

Pentru a răspunde la întrebări de acest gen, nu este suficient să avem informațiile necesare, acestea trebuie și organizate în mod corespunzător pentru a primi răspunsul într-un timp acceptabil.

**Reprezentarea informației** este **esențială** în domenii ca Știința calculatoarelor și Tehnologia informației.

# Introducere

---

Scopul programelor informatice nu este doar de a face calcule, ci și de a stoca și de a furniza informații, cât mai repede posibil.

Studiul structurilor de date și al algoritmilor face parte din nucleul științei calculatoarelor și tehnologiei informației

# Introducere

---

De multe ori o problemă poate fi abordată în mai multe moduri. Cum alegem între ele?

Obiective:

1. Să se proiecteze un algoritm care este **ușor** de înțeles, de implementat și de depanat
2. Să se proiecteze un algoritm care folosește în mod **eficient** resursele sistemului de calcul

Acest curs se axează pe îndeplinirea celui de-al doilea obiectiv.

Nevoia de a programa **eficient** nu dispare o dată cu evoluția hardware-ului. Chiar dacă avem sisteme de calcul tot mai performante, acestea sunt folosite pentru a rezolva probleme tot mai complexe, astfel nevoie de eficiență crește, nu scade.

# Noțiunea de structuri de date

---

La modul cel mai general, **o structură de date** este o **reprezentare a datelor** la care **se asociază o serie de operații**.

Ex. int, float în C/C++, o listă, un tablou

Pe fiecare din aceste structuri sunt definite **operații specifice**

Putem avea aceeași operație definită pe mai multe structuri de date, dar timpul de execuție este de cele mai multe ori diferit.

Spunem că o soluție este **eficientă** dacă rezolvă problema respectând constrângerile de folosire a resurselor impuse.

Exemple de **constrângeri**: de memorie, de timp, de cost, etc.



# Noțiunea de structuri de date

---

## **Greșeli** frecvente:

- Folosirea unor structuri de date cu care programatorul este familiar, fără a face o analiză în prealabil => program cu timpi mari de execuție
- Îmbunătățirea performanțelor unui program care deja respectă constrângerile legate de folosirea resurselor => complicarea nejustificată a codului

# Noțiunea de structuri de date

---

Când **alegem** structurile de date:

1. Analizăm problema pentru a determina operațiunile principale care trebuie suportate
2. Cuantificăm constrângerile legate de utilizarea resurselor
3. Selectăm acele structuri de date care respectă cerințele determinate la punctele anterioare

O abordare **centrată pe date** presupune:

1. Alegerea structurilor de date potrivite
2. Reprezentarea corectă a datelor folosind aceste structuri
3. Implementarea acestora într-un limbaj de programare concret

# Noțiunea de structuri de date

---

Cum ne influențează alegerea structurilor de date?

- Fiecare structură de date are avantaje și dezavantaje în termeni de cost (memorie, timp, etc.)
- În practică nu există o structură de date care să domine celelalte structuri în toate situațiile
- Fiecare structură de date necesită:
  - un anumit **spațiu** de memorie pentru a fi stocată
  - un anumit **timp** pentru a efectua operațiile de bază (inserție, căutare, ștergere, etc.)
  - un anumit **efort** pentru a fi implementată

# Noțiunea de algoritm

---

**Problemă** = o sarcină care trebuie indeplinită

De obicei este definită în termeni de date de intrare – date de ieșire ( input – output )

O problemă poate avea asociate **constrângeri** legate de resursele ce pot fi folosite (incluzând resurse de timp și memorie)

Se trece la găsirea unei soluții **doar după** ce problema a fost **bine definită** și clar **înțeleasă**!

Găsirea soluției se face cu ajutorul unui algoritm. Prin **algoritm** putem înțelege o serie de instrucțiuni pas cu pas care rezolvă o problemă dată

# Noțiunea de algoritm

---

**Problema** poate fi văzută ca o **funcție matematică**

O funcție matematică face legătura între datele de intrare și rezultate

F: Input-> Output

Găsirea unei funcții/proceduri prin care să mapăm Rezultatele cu Datele de intrare

Datele de intrare = parametrii funcției

Un set specific de valori reprezintă o instanță a problemei

Ex. Sortarea unor valori. Parametrii funcției pot fi un tablou de intregi și numărul de elemente.

Obs: Pentru același set de date de intrare rezultatul trebuie să fie mereu același!

# Noțiunea de algoritm

---

**Un algoritm** = o metodă sau un proces de a rezolva o problemă.

O metodă prin care mapăm datele de ieșire (rezultatele corecte) cu datele de intrare

Dacă problema este văzută ca o funcție, atunci algoritmul este implementarea (matematică) acelei funcții

F: Input -> Output

O problemă poate fi rezolvată prin mai mulți algoritmi (ex. sortarea unor elemente)

Avantajul cunoașterii mai multor algoritmi este că unul dintre aceștia poate fi mai eficient pentru o anumită instanță a problemei decât ceilalți

De exemplu un algoritm de sortare poate fi mai eficient când sortăm un număr mai redus de elemente și altul când sortăm un număr mare de elemente

# Noțiunea de algoritm

---

## Un algoritm trebuie să respecte următoarele cerințe

- Trebuie să fie **corect** – trebuie să furnizeze datele de ieșire corecte, pentru toate datele de intrare
- Trebuie să fie format dintr-o **serie de pași** concreți – implementabili pe sistemul de calcul pe care va rula acest algoritm. Aceștia trebuie să ruleze într-un timp finit
- Trebuie să **nu** fie **ambiguu**
- Trebuie să fie compus dintr-un număr **finit** de pași
- Trebuie să se **termine**

# Noțiunea de algoritm

---

**Program** = poate fi văzut ca o instanță sau o reprezentare concretă a unui algoritm într-un limbaj de programare

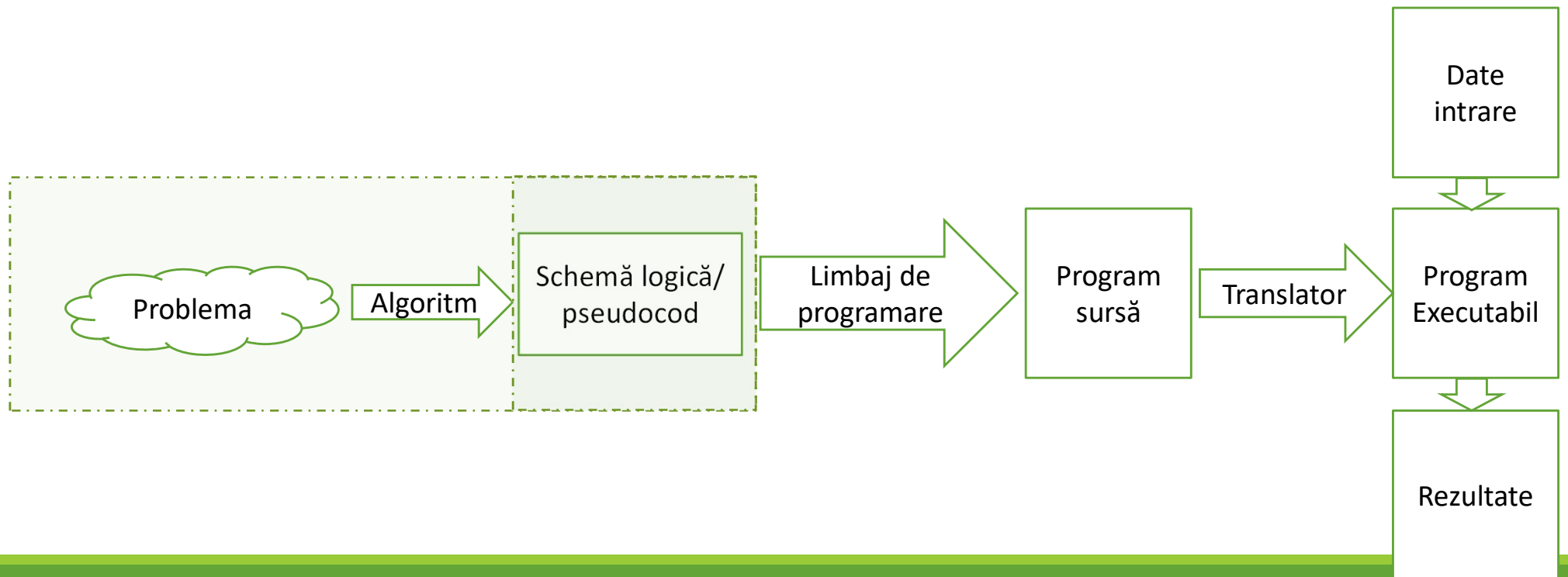
În mod evident pot exista mai multe programe care reprezintă diferite instanțe ale aceluiași algoritm

Un algoritm trebuie să se termine => nu orice program reprezintă o implementare a unui algoritm (ex. sistemele de operare)



# Scheme logice

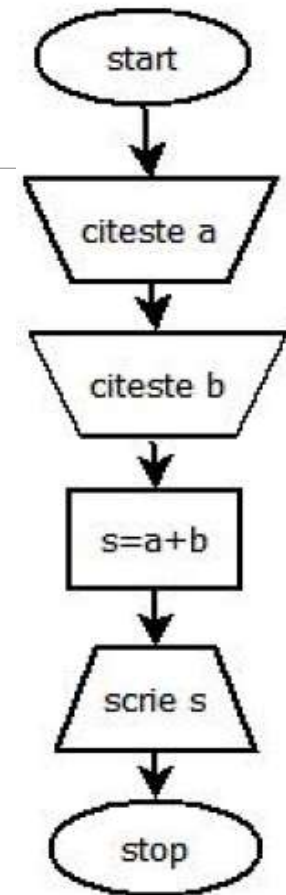
---



# Scheme logice



- **Schemele logice** sunt notații grafice care descriu grafic pașii unui algoritm
- În partea dreaptă se află un exemplu pentru calculul sumei a două numere



# Scheme logice

---

## **Datele din schemele logice**

- Variabile = zone de memorie care își schimbă valoarea și care se caracterizează printr-un nume

Ex. Suma, a, b

- Constante explicite = zone de memorie cu valori fixe, dar fără nume

Ex. 1, 7, 3.14

- Constante simbolice = zone de memorie cu valori fixe și cu nume

Ex. PI, e

# Scheme logice

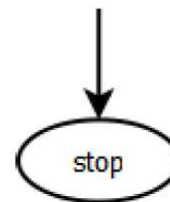
---

**Blocurile din schemele logice** - se referă la operațiile de bază

- De start



- De stop

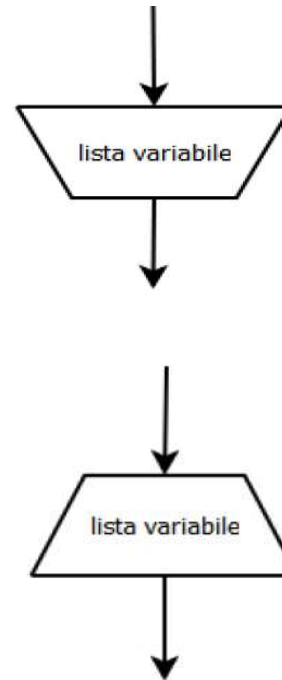


# Scheme logice

---

## Blocurile din schemele logice

- De intrare (de citire)
- De ieşire (de scriere)

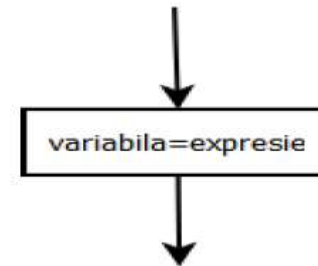


# Scheme logice

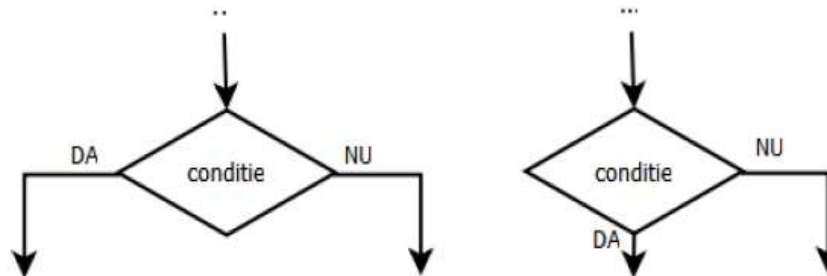
---

## Blocurile din schemele logice

- De atribuire



- De decizie



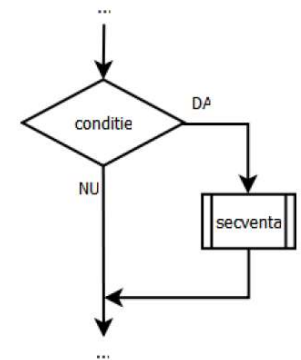
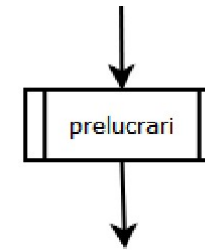
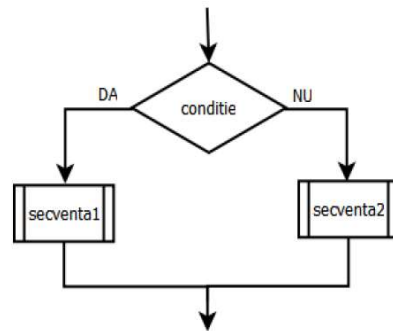
# Scheme logice

---

## Structurile de control – conțin mai multe blocuri

- Secvența

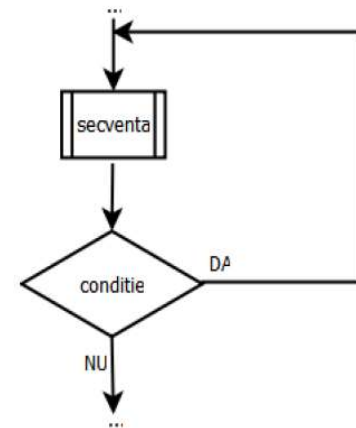
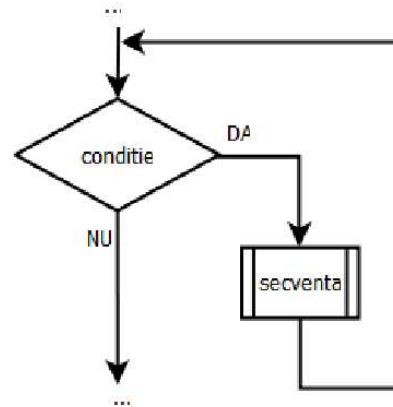
- Selecția



# Scheme logice

## Structurile de control

- Ciclul cu test inițial
- Ciclu cu test final

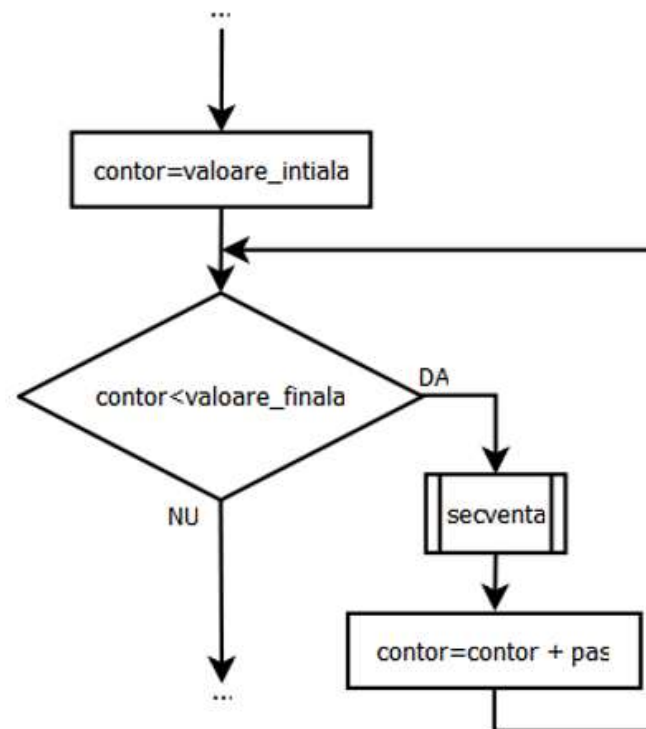




# Scheme logice

## Structurile de control

- Ciclul cu contor

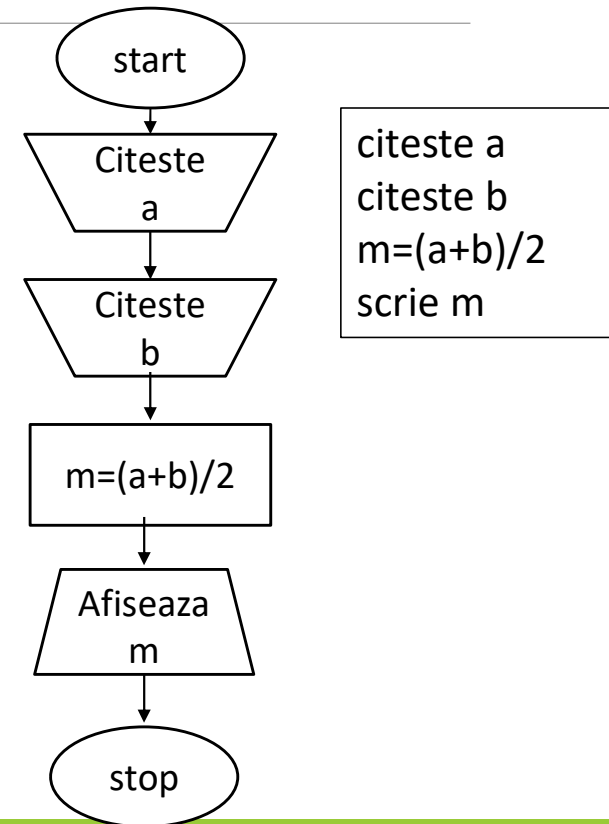


# Scheme logice

## Exemplu:

Media a două numere

Cum s-ar implementa în C o soluție bazată pe schema logică din dreapta?

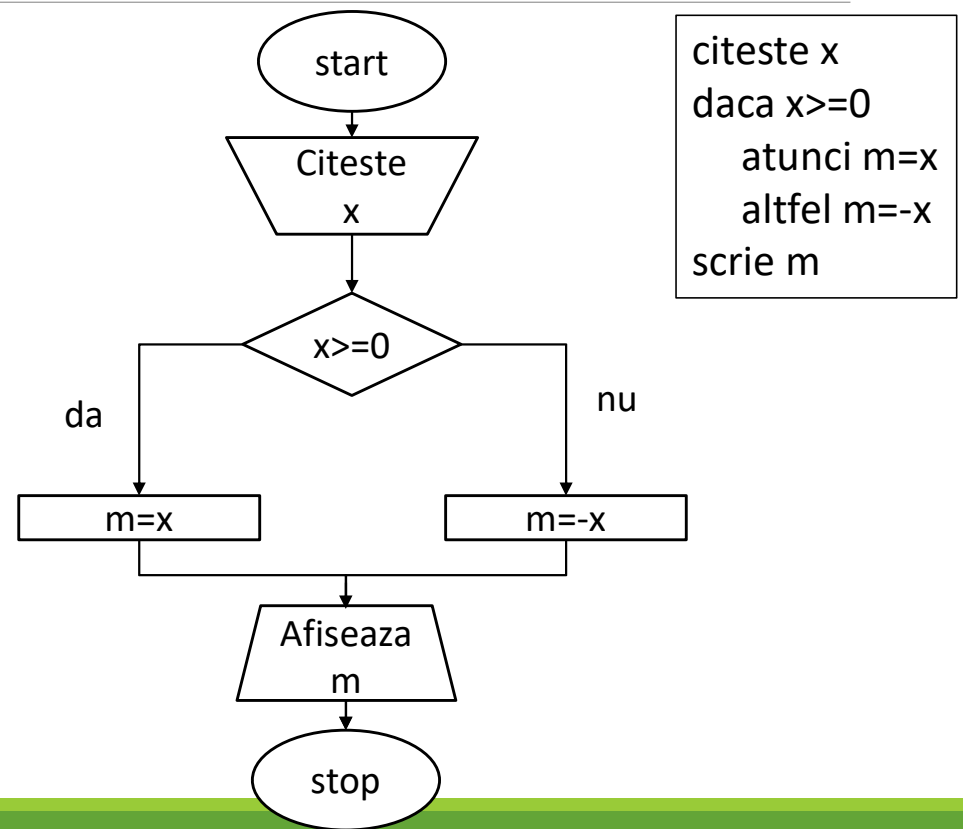


# Scheme logice

## Exemplu:

Funcția modul

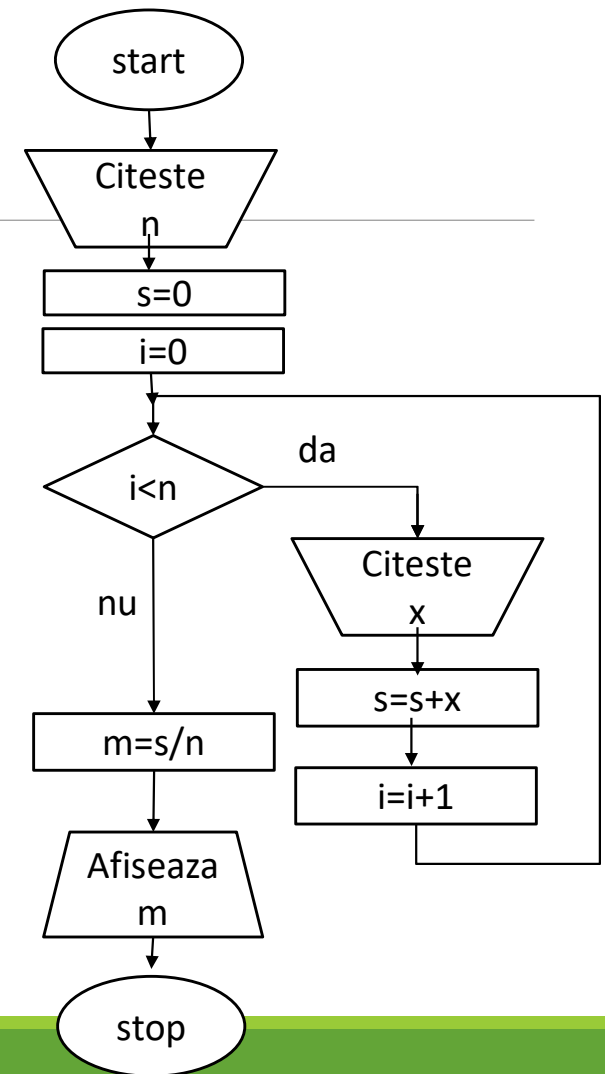
Cum s-ar implementa în C?



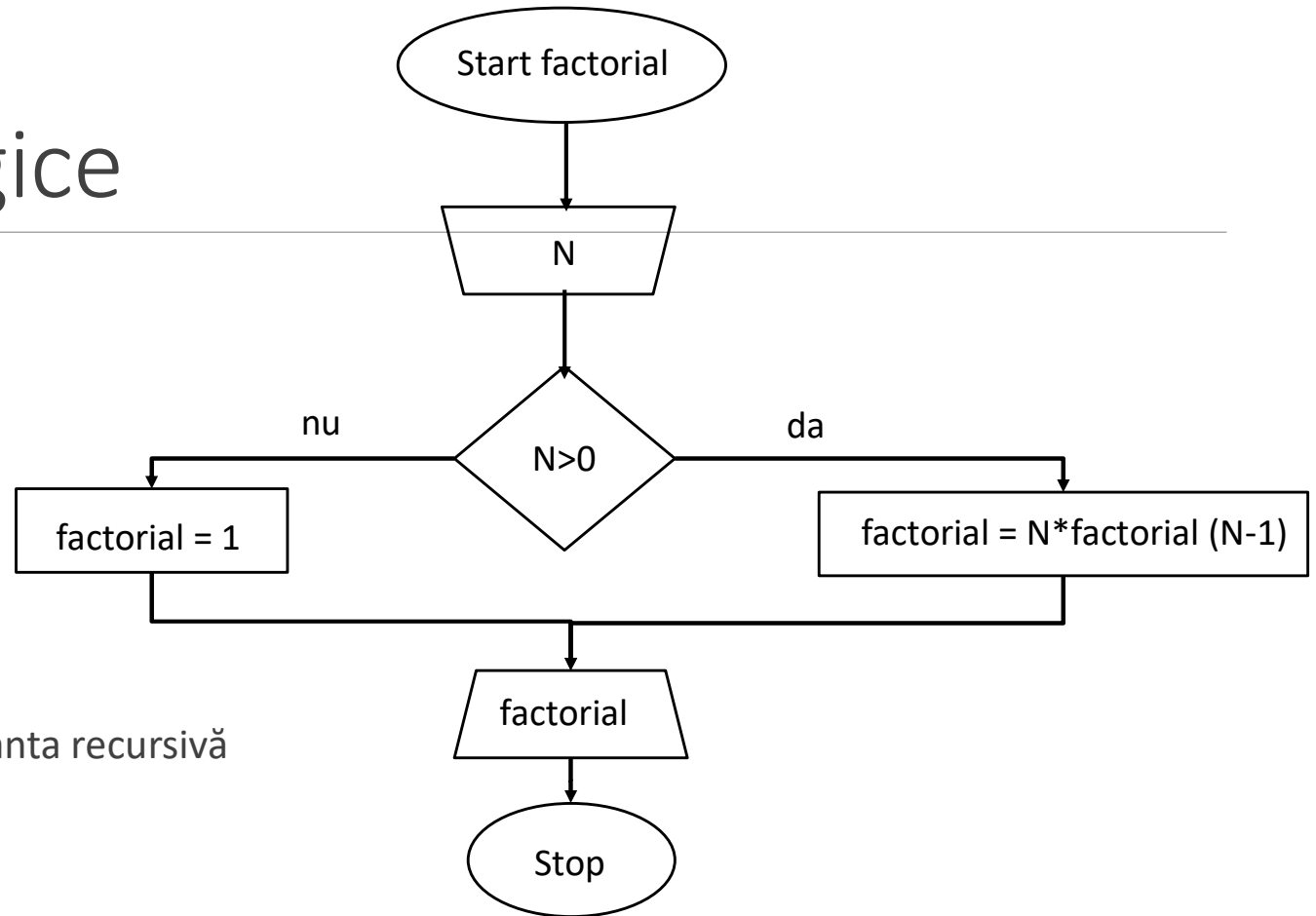
# Scheme logice

**Exemplu:**  
Media a  $n$  numere

```
citeste n
s=0
i=0
cat timp i<n
    citeste x
    s=s+x
    i=i+1
m=s/n
scrie m
```



# Scheme logice



## Exemplu:

Funcția factorial, varianta recursivă

# Exerciții

---

Ex1: Să se propună o schemă logică pentru funcția putere(baza, exponent), pentru numere naturale. Funcția returnează valoarea baza la puterea exponent

Ex2: Să se propună o schemă logică pentru o funcție care determină dacă un an este bisect

Ex3: Pe baza schemelor logice determinate anterior pentru exercițiile Ex1, Ex2 să se propună câte o variantă de implementare in C

# Bibliografie selectivă

---

- Minea, M., Limbaje de programare, materiale de curs  
<http://staff.cs.upt.ro/~marius/curs/lp/index.html>
- Chirila, C., Algoritmi, scheme logice si pseudocod  
<http://staff.cs.upt.ro/~chirila/teaching/upt/mpt11-upc/curs/upc12.pdf>
- Crețu, V. Structuri de date și algoritmi, Editura Orizonturi Universitare Timișoara, 2011
- Shaffer, C., “Data Structures and Algorithm Analysis”, 2012