# ALGORITMI

Noțiuni introductive

# 1.Noţiunea de algoritm.

**Def**. *Prin algoritm se înțelege o metodă de soluționare a unei clase de probleme, reprezentată de o succesiune finită de operații bine definite, numite instrucțiuni*.

Termenul de algoritm poate  înțeles în sens larg nefiind neapărat legat de rezolvarea unei probleme cu caracter științific, ci doar pentru a descrie într-o manieră ordonată activități care constau în parcurgerea unei succesiuni de pași (cum este de exemplu utilizarea unui telefon public sau a unui bancomat).

# Exemple

Unul dintre primii algoritmi se consideră a fi algoritmul lui Euclid (utilizat pentru determinarea celui mai mare divizor comun a doua numere naturale).

În matematică există o serie de algoritmi:

- cel al rezolvării ecuației de gradul doi,
- algoritmul lui Eratostene (pentru generarea numerelor prime mai mici decât o anumita valoare),
- schema lui Horner (pentru determinarea câtului și restului împărțirii unui polinom la un binom)

# Soluționarea problemei

Soluția problemei se obține prin *execuția* algoritmului.

Algoritmul poate fi  executat pe o mașină formală (în faza de proiectare și analiză) sau pe o mașină fizică (calculator) după ce a fost codificat într-un limbaj de programare.

## 2. Caracteristicile unui algoritm

***Generalitate***. Un algoritm destinat rezolvării unei probleme trebuie să permită obţinerea rezultatului pentru orice date de intrare şi nu numai pentru date particulare de intrare.

***Finitudine***. Adică se termină după un număr finit de paşi, indiferent cât de mulţi.

***Rigurozitate***. Prelucrarile algoritmului trebuie specificate riguros, fără ambiguităţi. În orice etapă a execuţiei algoritmului trebuie să se ştie exact care este următoarea etapă ce va fi executată.

***Eficienţă***. Algoritmii pot fi efectiv utilizaţi doar dacă folosesc *resurse de calcul* în volum acceptabil.
Prin resurse de calcul se înţelege volumul de memorie şi timpul necesar pentru execuţie.

**Exemple**

1.*Nu orice problemă poate  fi rezolvată algoritmic.*

a. Fiind dat un număr **n** să se determine toţi divizorii săi naturali.

Pentru această problemă se poate scrie un algoritm foarte uşor.

b. Fiind dat un număr **n** să se determine toţi multiplii săi.

Pentru această problemă **nu** se poate scrie un algoritm deoarece nu cunoaştem un criteriu de oprire a operaţiilor.

2.*Un algoritm trebuie să funcţioneze pentru orice date de intrare.*

Fiind date numerele **a, b, c** să se afişeze maximul dintre ele.

O posibilă soluţie ar fi:

Dacă **a>b** și **a>c** se afişează **a**, altfel

Dacă **b>a** și **b>c** se se afişează **b**, altfel

Dacă **c>a** și **c>b** se se afişează **c**

Algoritmul nu funcţionează dacă 2 valori sunt identice şi de valoare maximă.

**Exemple**

*3. Un algoritm trebuie sa se oprească.*

Se consideră următoarea secvenţă de prelucrări:
Pas 1. Atribuie variabilei ***x*** valoarea 1;
Pas 2. Măreste valoarea lui ***x*** cu 2;
Pas 3. Daca ***x*** este egal cu 100 atunci se opreşte prelucrarea altfel se reia de la Pas 2.

Este usor de observat ca *x* nu va lua niciodată valoarea 100, deci succesiunea de prelucrări nu se termină niciodată. Din acest motiv secvenţa nu poate fi considerată un algoritm corect.

*4. Prelucrările dintr-un algoritm trebuie să fie neambigue (clare).*

Consideram următoarea secvenţă de prelucrări:
Pas 1. Atribuie variabilei ***x*** valoarea 0;
Pas 2. *Fie* se măreşte ***x*** cu 1 *fie* se micşorează ***x*** cu 1;
Pas 3. Daca ***x*** *aparţine* [-10*;* 10] se reia de la Pas 2, altfel se opreşte algoritmul.

## Exemple

*5. Un algoritm trebuie să se oprească după un interval rezonabil de timp.*

Fiind dat un număr **n** se cere să se determine de câte ori a apărut o cifră **c** în reprezentarea tuturor numerelor naturale mai mici ca **n.**

O rezolvare simplă ar fi să luăm toate numerele mai mici ca **n** şi să vedem de câte ori apare cifra **c** în fiecare dintre ele. Soluţia e simplă şi pentru valori mici ale lui **n** algoritmul se termină într-un interval de timp rezonabil, dar pentru valori mari timpul de terminare al algoritmului creşte nepermis de mult.

# PAŞII REALIZĂRII UNUI ALGORITM

1. Citirea cu atenţie a enunţului problemei.

2. Identificarea datelor de intrare şi a celor de ieşire.

3. Rezolvarea propriu-zisă a problemei pe cazuri particulare şi reprezentative. În acest moment **nu** se încearcă scrierea programului ci doar determinarea metodei de rezolvare, generalizarea şi înţelegerea acesteia.

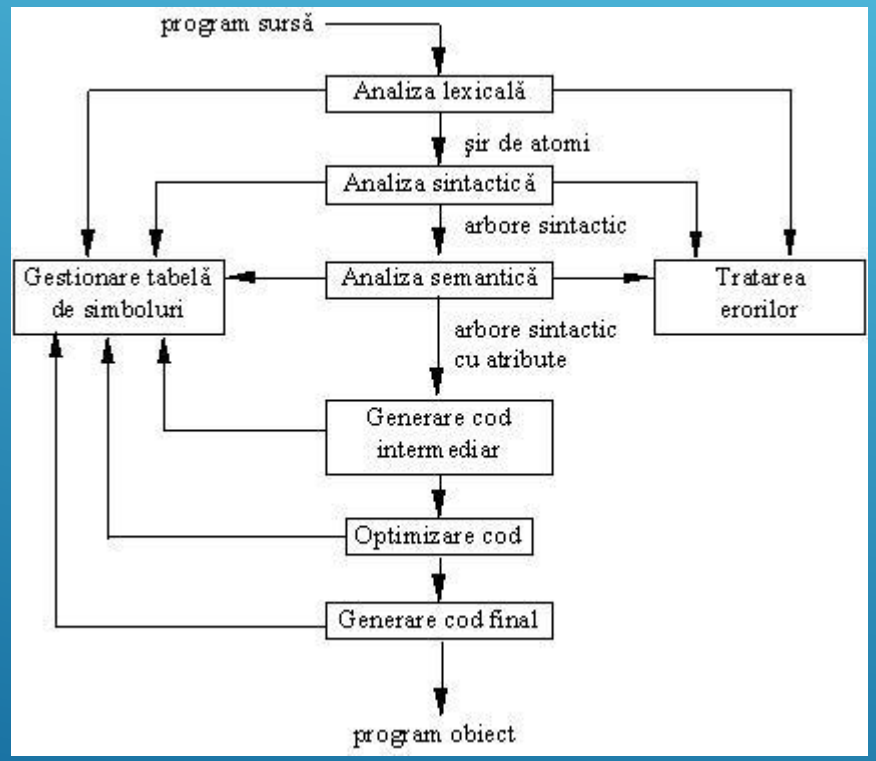
4. Descrierea în limbaj natural a soluţiei problemei.
Dacă nu sunteţi capabili să descrieţi metoda folosită în limbaj natural e puţin probabil să o puteţi face într-un limbaj de programare care e mai restrictiv decât limbajul natural.

5. Scrierea programului într-un limbaj de programare.

6. Testarea programului.
Testarea se face pe mai multe seturi de date care să acopere cazurile posibile ce pot apărea.

# Realizarea unui program

1. Scrierea programului folosind un mediu de dezvoltare (IDE)
2. Compilarea programului




3. Rularea programului obținut