II. Algoritmi

1. **Noţiunea de algoritm. Caracteristici. Exemple**
2. **Etapele rezolvarii unei probleme**
3. **Obiectele cu care lucrează algoritmii**
4. **Reprezentarea algoritmilor**

1. Noţiunea de algoritm

Algoritmul = metoda de solutionare a unui tip de probleme, constand intr-o multime finita, bine definita si ordonata de operatii.

Proprietati:

1) Generalitate –algoritmul rezolva o clasa de probleme nu o problema particulara

Ex : Nu 3+2 ci a+b

2) Claritate – algoritmul nu contine ambiguitati
3) Finitudine – algoritmul se termina dupa un numar finit de pasi

Alte proprietati:

- Completitudinea – algoritmul tine cont de toate cazurile particulare ale problemei generale. Ex : calculul lui 2 la n. Caz particular : 2 la 0 care trebuie tratat separat.
- Eficienta – algoritmul se va executa cu numar minim de pasi, folosind un minim de memorie
- Realizabilitatea – sa poata fi codificat intr-un limbaj de programare

Observatia1. Nu orice problema admite un algoritm de rezolvare. Ex: pot realiza un algoritm pentru determinarea numarului de divizori ai unui numar, dar nu pentru determinarea numarului de multiplii.
Observatia2. Doi agoritmi sunt echivalenti cand pentru aceleasi date de intrare se obtin aceleasi date de iesire.

Ex: Analogie cu "algoritmii" pe care ii executa omul

pas 1.Pune tigaia pe foc

pas 2.Pune o lingurita de ulei in tigaie

pas 3.**Cit timp** uleiul nu s-a incins asteapta

pas 4.Pune oualele in tigaie

pas 5.Astepta **pana cand** se rumenesc

pas 6.**Daca** nu tii regim, pune sare

Observam 3 tipuri de instructiuni:

- De tip executa
- Conditionale -cuvant cheie "daca"
- Repetitive –cuvinte cheie " cat timp" , "pana cand"

TEMA 1

Dati 2 exemple de algoritmi din viata de zi cu zi

Manual : pg 28/Ex. 1,2

2. Etapele rezolvarii unei probleme

Completitudine: Studiu de caz1:

Sa luam spre analiza, rezolvarea ecuatiei de gradul I : ax+b=0; Aceasta problema presupunea aflarea lui x, in functie de 2 date numerice, a si b.

Din enuntul problemei, deducem ca, a si b vor fi datele pe care noi le vom primi iar noi va trebui sa il aflam pe x in functie de acestea. Spunem ca ceea ce se da, a si b, sunt date de intrare, iar ceea ce se cere, adica x, reprezinta data de iesire.

Va trebui sa elaboram o metoda, prin care sa rezolvam in pasi aceasta problema. Un algoritm rapid care ar descrie modul de rezolvare ar fi urmatorul:

```
pas 1 : preluam datele de intrare a si b
pas 2 : calculam pe x ca fiind -b/a
pas 3: furnizam rezultatul x
```

Insa acest mod de rezolvare nu se va putea insa aplica pentru orice valori a si b. Putem sa observam, ca, in cazul in care a = 0, algoritmul nostru ar face o operatie nepermisa, anume impartire la 0 . Ca urmare, ar trebui sa gandim o solutie care sa ia in considerare si pacest caz particular. Vom distinge 2 astfel de cazuri:

a) cand a = 0 si b=0 , x va putea fi orice numar

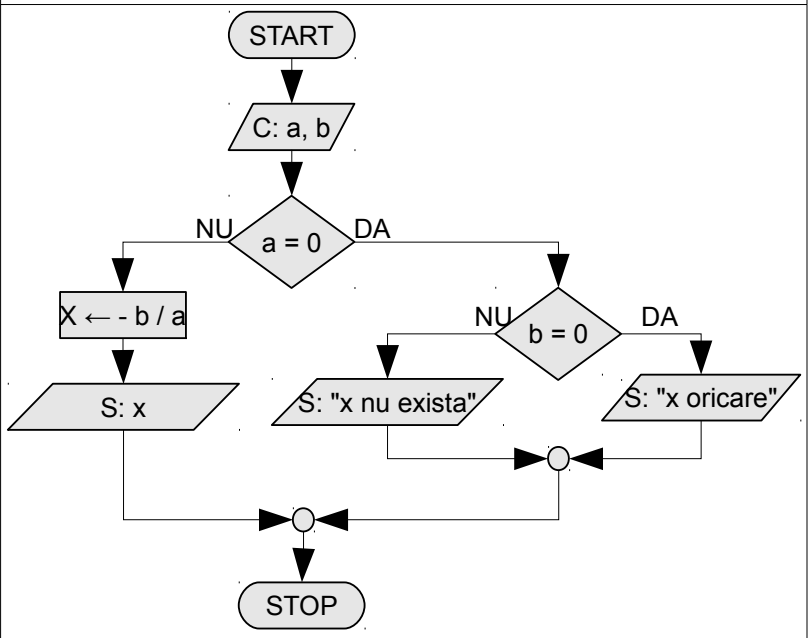
b) cand a=0 si b=0 ; in acest caz nu avem nici o solutie pentru x.

Tinand cont de aceste doua cazuri particulare, vom putea completa algoritmul, rezultand unul **complet** care va putea furniza un raspuns pentru orice pereche de valori a,b.

```
pas 1: preluam datele de intrare a si b
pas 2: daca a=0 si b=0 furnizam mesaj "exista o infinitate de solutii"
pas 3: daca a=0 si b <> 0 furnizam mesaj "nu esista nici o solutie"
pas 4: daca a <> 0 calculam  x =-b/a si furnizam rezultatul x
```

Programatorii au cazut de acord in privinta unor reguli de reprezentare a acestor pasi ai algoritmilor, astfel incat acestia sa fie scrisi cat mai schematic, reguli pe care le vom studia la capitolul "reprezentarea algoritmilor". Iata un exemplu de reprezentare a acestui algoritm:

Dupa ce am analizat, am definit pasii de rezolvare pentru problema data si am verificat algoritmul rezultat pe un set de date de intrare, putem sa implementam intr-un limbaj de programare algoritmul.

PSEUDOCOD	SCHEMA LOGICA
Intreg a,b Citeste a, b daca (a=0) atunci daca (b=0) atunci scrie “x oricare” altfel scrie “x nu exista” altfel x=-b/a scrie x	

Studiu de caz1:

Eficienta (timp mai scurt): Sa luam spre analiza, calculul sumei primelor n valori naturale;

Analizam 2 metode de rezolvare : prin formula sau prin adaugarea pe rand a celor n valori, la calculul sumei

Eficienta (memorie folosita mai mica): Suma elementelor dintr-o secventa de ne valori.

Iata pe scurt **etapele de rezolvare a unei probleme:**

pas 1. Identificarea **datelor** de intrare/iesire (ce se da si ce se cere?)

Ce sunt datele? Ce sunt informatiile?

pas 2. Proiectarea algoritmului

Cum reprezentam un algoritm pe hartie? Cum identificam cazurile particulare ale unei probleme?

pas 3. Testarea algoritmului dupa faza de proiectare.

pas 4. Traspunere algoritm intr-un limbaj de programare

Cum implementam in calcularor un algoritm ?

pas 5. Testarea algoritmului dupa implementarea acestuia (depistare erori de sintaxa sau erori logice)

Cum verificam algoritmul realizat ?

Tema 2:

Dati 2 exemple de algoritmi in care sa puneti in evidenta proprietatile algoritmului (Completitudine, Eficienta, etc)

3. Obiecte cu care lucreaza algoritmii

3.1 Date(variabile sau constante)

Pot fi date de doua tipuri:

- Elementare(intregi, reale, logice, caracter)
- Structurate

(optional citim despre memorarea numerelor naturale si intregi din documentul "Memorarea Datelor")

De asemenea datele pot fi:

- **Constante** = informatii care se autodefinesc; date care nu isi modifica valoarea;

Constante numerice: intregi: 2, 4; reale: 5.2, 7.8

Constante logice: TRUE, FALSE

Constante Caracter: 'c', '\n'

Constante sir de caractere : "mama are mere", "Cuvantul \"while\" reprezinta cuvant cheie pentru limbajul c "

- **Variabile** = date care isi modifica valoarea. Pot fi de aceleasi tipuri ca si constantele.

Proprietatile variabilelor: nume, tip, locatie de memorie, valoare la un moment dat

Obs: Algoritmul urmator va afisa de doua ori valoarea 3 deoarece in locatia de memorie a lui x ramane doar ultima valoare primita.

X=2

X=3

Scrie x

Scrie x

De adaugat o schema grafica cu proprietatile unei variabile

3.2 Operatori

Operatorii –au rolul de a preciza ce operatii se vor realiza asupra datelor cu care lucram. Operatorii se aplica doar anumitor tipuri de date. De pilda nu putem aduna doua caractere sau nu putem imparti cu rest numerele reale (cele cu virgula).

Tipuri de operatori:

3.2.1 Matematici :

+,-,*,/

In plus vom avea doi operatori :

div =catul impartirii a doua numere intregi

mod=restul impartirii a doua numere intregi

Ex: 2+3, a div 2, a+2*x; aceste expresii au un rezultat numeric, de aceea se numesc expresii aritmetice

3.2.2 Relationali: < ,>, >=,<=,=,<>

Ex: a<b, 2=3,

Aceste expresii au ca rezultat o valoare logica (adevarat sau fals). De aceea vom spune despre acestea ca sunt expresii logice chiar daca nu contin operatori logici.

3.2.3 Logici: AND/OR/NOT

a	b	a AND b	a OR b	NOT a
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

Proprietati (Relatiile lui Morgan):

1) NOT (a AND b)=NOTa OR NOTb
2) NOT (a OR b)=NOTa AND NOTb

Demonstratie pentru prima relatie:

a	b	a AND b	NOT(a AND b)	NOT a	NOT b	NOTa or NOTb
TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE

TEMA in clasa: Demonstrati relatia a doua a lui MORGAN urmand exemplul primei demonstratii.

a	b	a OR b	NOT(a OR b)	NOT a	NOT b	NOTa AND NOTb
TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE

3.3 Expresii

Ca si la matematica, o expresie este formata din operanzi si operatori. In functie de tipul de date, vor fi aplicati operatori specifici. De pilda , are sens sa aplicam un operator aritmetic asupra unor date numerice. Nu are sens o peratie de adunare asupra a doua date caracter.

Cum nu are sens de asemenea sa aplicam un operator logic asupra a 2 numere: ce sens ar avea expresia : “2 Si 3” ?

Expresiile se pot clasifica si ele in functie de valoarea rezultata in urma evaluarii expresiei. Vom avea :

1. **expresii intregi**. ex: 2+3; 4*50; 30+a, unde a este o variabila numerica de tip intreg
2. **expresii reale**. ex: 2.3+4, 2+x, unde x este o variabila numerica de tip real
3. **expresii logice**. Ex: a>b AND 4<c, c%2=1

Evaluarea unei expresii

Pentru evaluarea expresiilor se respecta regulile de baza invatate la matematica. Se evalueaza intai expresiile dintre parantezele rotunde, apoi se executa operatiile in ordinea prioritatii lor. Daca exista operatii cu aceasi prioritate, ele se executa in ordine, in functie de asociativitatea lor. Prioritatea 1 este considerata cea mai mare

Prioritate	Operatori	Simbol	Asociativitate
1	Negatia logica	NOT	De la stanga la dreapta
2	Aritmetici multiplicativi	*, /, MOD, DIV	De la stanga la dreapta
3	Aritmetici aditivi	+, -	De la stanga la dreapta
4	Relationali	<, >,≤, ≥, ≠, =	De la stanga la dreapta
5	Conjuntia logica (si)	AND	De la stanga la dreapta
6	Disjunctia logica (sau)	OR	De la stanga la dreapta

4. Reprezentarea algoritmilor

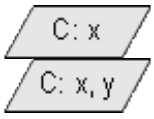
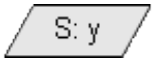
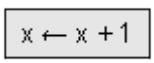
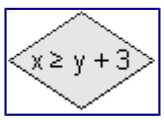
Obs: Algoritmul "oua ochiuri" este un algoritm pe care il va executa un om. Instructiunea "Pune tigaia pe foc" nu ar putea fi executata de catre calculator. De aceea algoritmii pe care ii vom studia la informatica vor contine doar operatii pe care un calculator le-ar putea executa. Aceste operatii pot fi:

- Operatii de intrare/iesire =operatiile de citire/scriere
- Operatii de atribuire
- Operatii decizionale

Vom folosi doua modalitati de reprezentare a algoritmilor:

1. **PSEUDOCOD** = limbaj apropiat limbajului nostru natural, dar care este de asemenea foarte apropiat si de limbajele de programare in care vor fi transpusi algoritmii.
2. **SCHEMA LOGICA** = Reprezentare grafica. Fiecărui tip de prelucrare elementară (fiecarei operatii) îi corespunde un simbol grafic. Prelucrările succesive sunt indicate prin conectarea prelucrărilor elementare şi prin săgeţi.

Obs. Asa cum un constructor nu se apuca de construit pana ce nu are un plan al constructiei, nici un programator nu trebuie sa se apuce de implementat algoritmul direct in calculator. El isi va reprezenta mai intai algoritmul pe hartie, concentrandu-se mai intai asupra metodei de rezolvare a problemei, asupra analizei cazurilor particulare si asupra modului de reprezentare .

Descriere	**Schema logica**	**Pseudocod**
Blocul de inceput/sfarsit - orice schema logica incepe cu un bloc de start si se termina cu blocul de stop		
Blocul de citire - se citesc de la dispozitivul de intrare valorile variabilelor specificate in lista_variabile (separate prin virgula)		**Citeste *lista_variabile*** Citeste x Citeste x, y
Blocul de scriere (doua variante) - se scriu la dispozitivul de iesire valorile obtinute in urma evaluarii expresiilor din lista(separate prin virgula)		**Scrie *lista_expresii*** *Scrie y*
Blocul de atribuire - se evalueaza expresia; valoarea obtinuta este memorata in variabila, vechea valoare pierzandu-se;		**Variabila←expresie** x←x+1
Blocul de decizie - se evalueaza conditia: daca e adevarata se continua cu prelucrarea indicata de ramura ***da***, altfel cu ramura ***nu***; conditia poate contine operatori relationali, operatori logici		**daca** (conditie) ... daca (x ≥ y + 3)...

Operatii pe care le efectueaza algoritmii

1) Operatii de intrare iesire ✳Citire/scriere

Operatia de citire este operatia prin care se preiau date de la un dispozitiv de intrare(ex. De la tastatura- de la utilizator)

Operatia de scriere este operatia prin care sa preiau date din memoria interna a calculatorului si se transfera catre un dispozitiv de iesire (ex: Catre monitor - catre utilizator)

Aplicatia1: Se citesc 2 valori intregi a si b de la tastatura. Sa se afiseze produsul acestora pe ecran.

Pseudocod	**C**
Intreg a,b Citest a,b Scrie "produsul numererlor este", a*b	**#include <stdio.h>** **#include <stdlib.h>** **int main(){** **int a, b;** **printf("Introduceti doua numere: ");** **scanf("%d", &a);** **scanf("%d", &b);** **printf("Produsul numerelor este %d\n", a * b);** **return 0;** **}**

2) Operatii de atribuire

Operatia de atribuire este operatia prin care dam valoarea unei variabile.

Aplicatia2: sa se interschimbe continutul a doua variabile.

Pseudocod	**C**

Intreg a,b,aux
Citeste a,b
Auxa
A<-- b
B<-- aux
Scrie a, b
	#include <stdio.h>
	#include <stdlib.h>
	int main(){
	int a, b, aux;
	printf("Introduceti doua numere: ");
	scanf("%d", &a);
	scanf("%d", &b);
	aux=a;
	a=b;

	b=aux;
	printf("Valorile interschimbate sunt %d %d:\n", a, b);
	return 0;
	}

3) Operatii decizionale

Operatia decizionala este operatia prin care calculatorul poate decide valoarea de adevar a unei expresii logice. In functie de aceasta operatie calculatorul poate executa sau nu alte operatii.

Ex3: Se citesc 2 numere. Sa se afiseze care este numarul cel mai mare

Pseudocod	**C**

Intreg a,b,aux
Citeste a,b
daca(a<b) atunci scrie b
altfel scrie a
	#include <stdio.h>
	#include <stdlib.h>
	int main(){
	int a, b ;
	printf("Introduceti doua numere: ");
	scanf("%d", &a);
	scanf("%d", &b);
	if(a<b)
	printf("max este %d:\n", b);
	else
	printf("max este %d:\n", a);
	
	return 0;
	
	}

Tema:

1. Scrieti schema logica pentru aplicatiile 1,2,3 de mai sus.
2. Stiti o alta metoda prin care puteti interschimba continutul a doua variabile fara a utiliza o a treia ? Daca da, scrieti algoritmul.
3. Se citesc 3 numere. Sa se afiseze valoarea cea mai mare.
4. Se citesc 4 numere. Sa se afiseze valoarea cea mai mare.
5. Ionel doreste sa cumpere 2 produse de la supermarket. Insa fiecare dintre cele doua produse sunt in doua variante de pret. Cunoscandu-se cele 2 preturi pentru fiecare produs, sa se afiseze cat ar economisi Ionel daca ar cumpara produsele cele mai ieftine.