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 2,3

Pseudocod și Bazele logicii binare

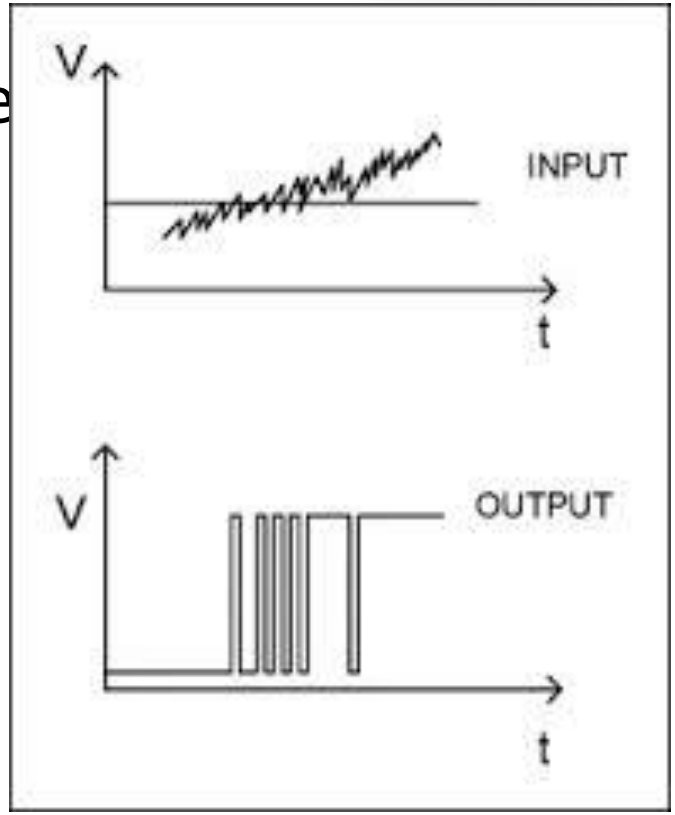
Dr. ing. Kiss Istvan

istvan.kiss@umfst.ro

Cuprins

1. Bazele logicii binare

- Digital? -> se referă la ceva care are la bază cifre, în special cifre binare
- Un sistem de calcul modern folosește cifre binare (biți) în reprezentarea datelor.
- Un bit are valoare 1 sau 0
- Fizic:
  - Tensiunea de 5V este 1 logic
  - Tensiunea de 0V este 0 logic




```
'B' =  0 1 0 0 0 0 1 0
    =  L H L L L L H L
    =
```

1. Bazele logicii binare

- Logica binară pornește de la premiza că o propoziție poate avea numai unul din două rezultate posibile: **adevărat** sau **fals.**
- Adevărat și fals sunt abstractizate cu valorile binare 1 și 0.

$$\underline{234}:2=\underline{117}:2=\underline{58}:2=\underline{29}:2=\underline{14}:2=\underline{7}:2=\underline{3}:2=\underline{1}:2=0$$

0 1 0 1 0 1 1 1

Numărul $243_{(10)}$ este $1110\ 1010_{(2)}$

2. Operații de bază

- ”1” – un bit
- ”1001” – cuvânt (număr) binar de 4 biți
- 2 tipuri
  - 1 && 0 = ?
  - 10_{10} & 2_{10} = ?

Operații logice binare de bază						
Operație	**Simbol logic**	**Simbol programare (logică)**	**Simbol programare (binar)**	**Operand 1**	**Operand 2**	**Rezultat**
AND (și)	∧	&&	&	0	0	0
				0	1	0
				1	0	0
				1	1	1
OR (sau)	∨	||	|	0	0	0
				0	1	1
				1	0	1
				1	1	1
NOT (negație)	¬	!	~	-	1	0
				-	0	1

3. Propoziții logice

- Se formează pornind de la operații de bază
- Dacă P, Q sunt doi operanzi logici:
- P v Q este echivalent cu propoziția: P sau Q, având valoarea de adevăr 1 dacă măcar unul dintre elementele P sau Q este evaluat ca 1 sau 0 (fals) altfel.
- De ex.: $(P \vee Q) \wedge \neg(P \wedge Q)$ este 1 dacă și numai dacă una dintre P și Q este 1 și cealaltă 0.

- Ex.: Să spunem că **P** înseamnă "ai 14 ani sau mai mult" iar **Q** înseamnă "nu ai buletin". Neștiind nici dacă aveți 14 ani și nici dacă v-ați făcut deja buletin, propoziția logică "$R = P \wedge Q$", unde **R** înseamnă "ar trebui să vă faceți buletinul" este validă.

3. Propoziții logice

a > b	a <= b
a < b	a >= b
a == b	a != b

(a>b) && (c>a), atunci (c>b)?

3. Propoziții logice - probleme

- **Argumente**:

a) Daca autobuzul pleaca la ora fixata si nu are întârzieri pe traseu, înseamna ca va ajunge la timp. Întrucât autobuzul nu a ajuns la timp, rezulta ca el nu a plecat la ora fixata sau ca a avut întârzieri pe traseu.

b) Daca populatia creste în progresie geometrica, în timp ce resursele cresc în progresie aritmetica, saracia generalizata este inevitabila. Populatia nu creste în progresie geometrica. Deci, saracia generalizata nu este inevitabila.

c) Daca primarul ales este un bun gospodar sau dispune de consilieri priceputi, atunci fondurile vor fi directionate spre modernizarea utilitatilor publice. Cum fondurile sunt destinate modernizarii utilitatilor publice, înseamna ca primarul ales este un bun gospodar sau dispune de consilieri priceputi si onesti.

- **Cerinte**:

Identificati propozitiile componente.

4. Pseudocod.

- Pseudocodul este un limbaj format din cuvinte cheie dintr-o limbă naturală și operatori cunoscuți din matematică

- Cuvintele cheie sunt echivalente cu instrucțiuni ale limbajului de programare

- Pseudocodul trebuie să fie independent de limbajul de programare

4. Forma generală

- Date de intrare …
- Rezultate …
- Algoritmul Nume este
- ....
- Sfârșit algoritm

Cuvinte cheie

- Date, variabile, expresii, atribuire
- Atribuirea se notează := sau ←
- Citire, scriere
- Ramificație/Decizie – structură alternativă
- Structuri de algoritmi/programare
  - Liniară, alternativă, repetitivă/ciclică

Date

- Date de intrare, interne și de ieșire
- Tipuri generice
  - **Natural** *10*
  - **Întreg** *-6*
  - **Real** *-53.6354*
  - **Caracter** *‘A’, spațiu*
  - **Șir de caractere** *“Algoritmi fundamentali”*
  - **Șir de natural** *[5, 25, 30, 5000]*
  - **Logic** *adevărat/fals (1 sau 0)*

5. Noțiunea de variabilă

- Variabilele în algoritmică și programare sunt date ale căror valoare se modifică pe parcursul execuției algoritmului.
- Se utilizează pentru reținerea datelor de intrare, interne și de ieșire.
- Variabila are: denumire, tip, valoare inițială.
- Exemple:
  - Natural contor:=0;
  - Întreg suma:=100;
  - Șir de caractere mesaj:=“Helo World”;

Operatori

	Operator	Semnificaţie
Operatori aritmetici	+	Adunare
	-	Scădere
	*	Înmulţire
	/	Împărţire
Operatori relaţionali	<	Mai mic
	<=	Mai mic sau egal
	>	Mai mare
	>=	Mai mare sau egal
	=	Egal
	<>	Diferit
Operatori logici	not	Negaţie
	si	Şi (conjuncţie)
	sau	Sau (disjuncţie)

6. Expresii

- O expresie este o înșiruire de operanzi și operatori
- Operandul poate fi variabilă sau constantă
- Expresie aritmetică: 3*x-6/y+5
- Expresie logică: a>b; (a>b) și (b>c) sau (c<d);

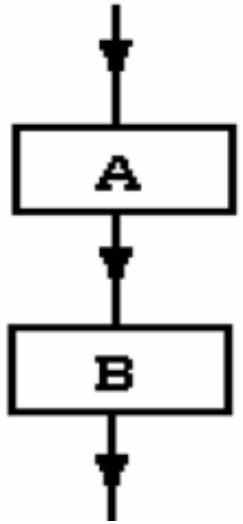
7. Atribuire

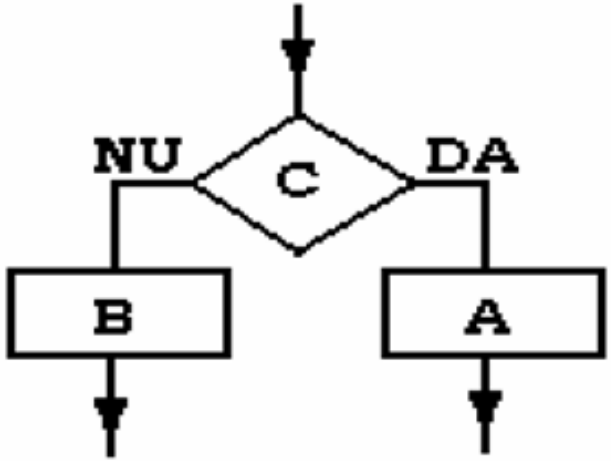
- Prin atribuire se reține într-o variabilă o valoare
- Variabilă:=expresie;

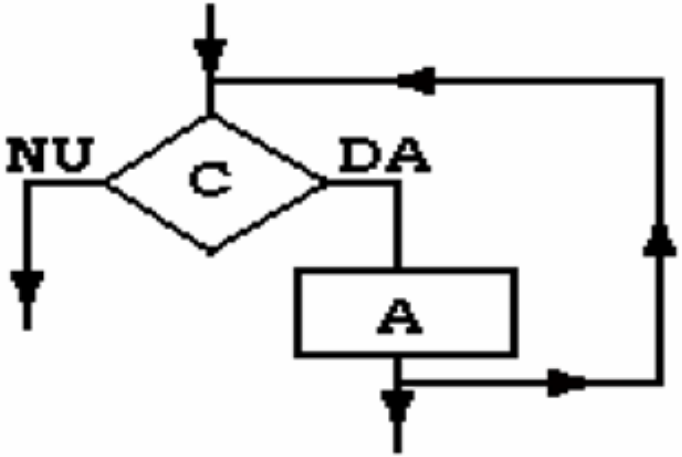
- Exemple:
  - Real pi:=3.14;
  - Intreg a:=15;
  - Real r:=2*15+35+45/2;
  - Intreg a:=b;

8. Structuri de algoritmi

1. Liniară (secvențială)
2. Alternativă
3. Repetitivă







9. Structură liniară

ALGORITMUL *VITEZA* ***ESTE****:*

CITEŞTE *D,T;*

V:=D/T;

TIPĂREŞTE *V;*

SF. ALGORITM

10. Structură alternativă (decizia)

Dacă *condiție* **atunci**

Instrucțiune1;

Instrucțiune2;

...

Instrucțiune n;

Sf. **Dacă**

Ex:

Dacă *variabila > 15* **atunci**

x:=x+1;

i:=0;

Altfel

x:=x+2;

Sf. **Dacă**

Dacă *condiție* **atunci**

Instrucțiune1;

Instrucțiune2;

...

Instrucțiune n;

Altfel

Instrucțiune1;

Instrucțiune2;

...

Instrucțiune n;

Sf. **Dacă**

10. Structură alternativă (decizia)

SELECTEAZĂ *i* ***DINTRE***

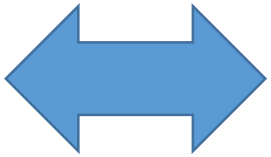
v1: A1;

v2: A2;

. . .

vn: An

SFSELECTEAZĂ



DACĂ *i=v1* ***ATUNCI*** *A1 ALTFEL*

DACĂ *i=v2* ***ATUNCI*** *A2 ALTFEL*

. . .

DACĂ *i=vn* ***ATUNCI*** *An* ***SFDACĂ***

. . .

SFDACĂ

SFDACĂ

10. Structură alternativă (decizia)

```
ALGORITMUL ECGRDOI ESTE:     {Rezolvarea ecuaţiei de gradul doi }
  CITEŞTE a,b,c; { a,b,c = Coeficienţii ecuaţiei }
  delta:=b*b-4*a*c;
  DACĂ delta<0 ATUNCI ind:=0;  { rădăcini complexe }
        r:=radical din (-delta);
        x1:=-b/(a+a);
        x2:=r/(a+a);
   ALTFEL ind:=1;          { rădăcini reale }
        r:=radical din delta;
        x1:=(-b-r)/(a+a);
        x2:=(-b+r)/(a+a);
  SFDACĂ
  TIPĂREŞTE ind, x1,x2;
SFALGORITM
```

11. Structură repetitivă

- Condiționată anterior
  - Cât timp ... Exec – *număr necunoscut de pași*
  - Pentru ... Exec – *număr cunoscut de pași*
- Condiționată posterior
  - Repetă ... Până când

11.1. cât timp ... Exec.

- ***CÂTTIMP*** *cond* ***EXECUTĂ*** *A* ***SF.CÂT***

```
întreg numar;
citire numar;
cat timp numar<>0 executa
    citire numar;
sf. cat timp
```

```
întreg numar;
natural contor:=1;
cat timp contor<=10 executa
    citire numar;
    contor:=contor+1;
sf. cat timp
```

11.1. cât timp ... Exec.

- ***CÂTTIMP*** *cond* ***EXECUTĂ*** *A* ***SF.CÂT***

ALGORITMUL *Euclid* ***ESTE****:* *{A3: Cel mai mare divizor comun}*

CITEŞTE *n1,n2;* *{Cele două numere a căror divizor se cere}*

d:=n1; i:=n2;

CÂTTIMP *i≠0* ***EXECUTĂ***

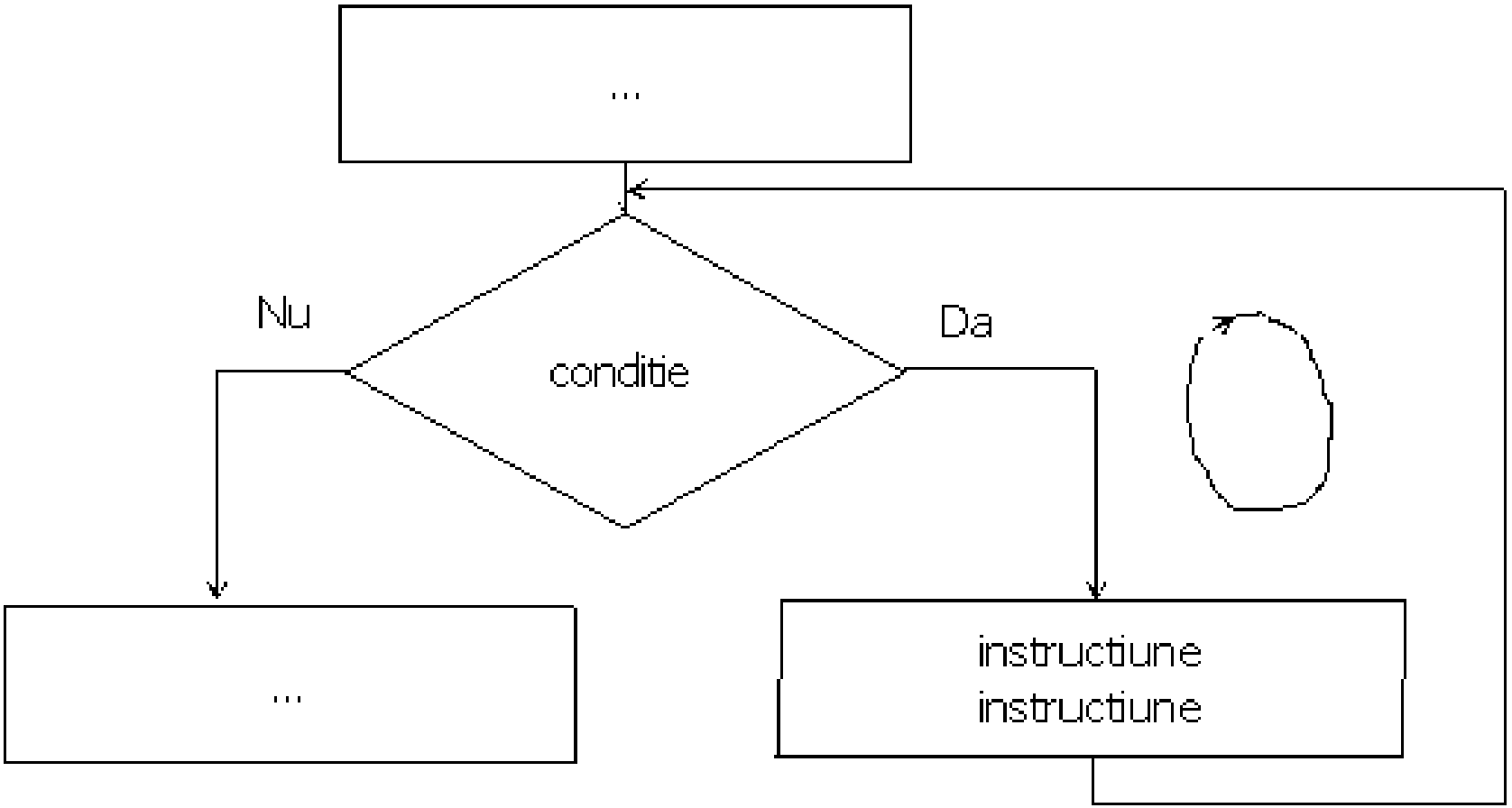
r:=d modulo i; d:=i; i:=r

SFCÂT

TIPĂREŞTE *d;* *{ d= cel mai mare divizor comun al }*

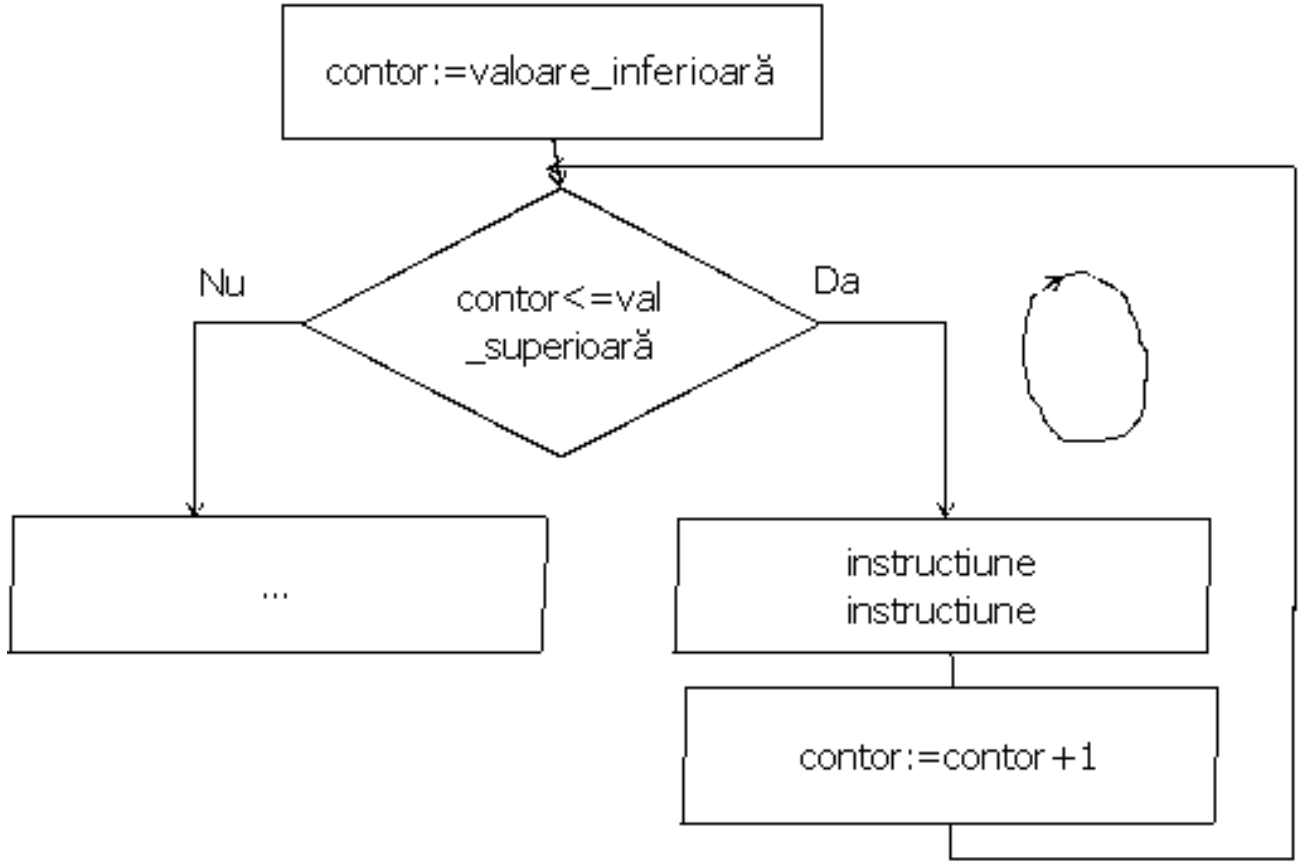
SFALGORITM *{ numerelor n1 şi n2 }*

11.1. Schemă logică cât timp



11.2. Pentru ... Exec.

- ***PENTRU*** *c:=l_i;l_f[;p]* ***EXECUTĂ*** *A* ***SF.PENTRU***



11.2. Pentru ... Exec.

- ***PENTRU*** *c:=l_i ;l_f [;p]* ***EXECUTĂ*** *A* ***SF.PENTRU***

ALGORITMUL *MAXMIN* ***ESTE*** *{ Algoritmul 5: Calculul } { valorii minime şi maxime }*

CITEŞTE *n,(x_i,i=1,n);*

valmin:=x_1; valmax:=x_1;

PENTRU *i:=2,n* ***EXECUTĂ***

DACĂ *x_i<valmin* ***ATUNCI*** *valmin:=x_i* ***SFDACĂ***

DACĂ *x_i>valmax* ***ATUNCI*** *valmax:=x_i* ***SFDACĂ***

SFPENTRU

TIPĂREŞTE *valmin,valmax;*

SFALGORITM

11.2. Pentru ... Exec.

- ***PENTRU*** *c:=l_i;l_f[;p]* ***EXECUTĂ*** *A* ***SF.PENTRU***

```
întreg numar;
natural contor;
pentru contor:=1..10 execută
    citire numar;
sf. pentru
```

11.2. Pentru ... Exec.

- ***PENTRU*** *c:=l_i;l_f[;p]* ***EXECUTĂ*** *A* ***SF.PENTRU***

Structura echivalentă este:

c:=l_i; final:=l_f;

REPETĂ

A

c:=c+p

PÂNĂCÂND *(c>final şi p>0) sau (c<final şi p<0)*

SFREP

11.3. repetă ... Până când

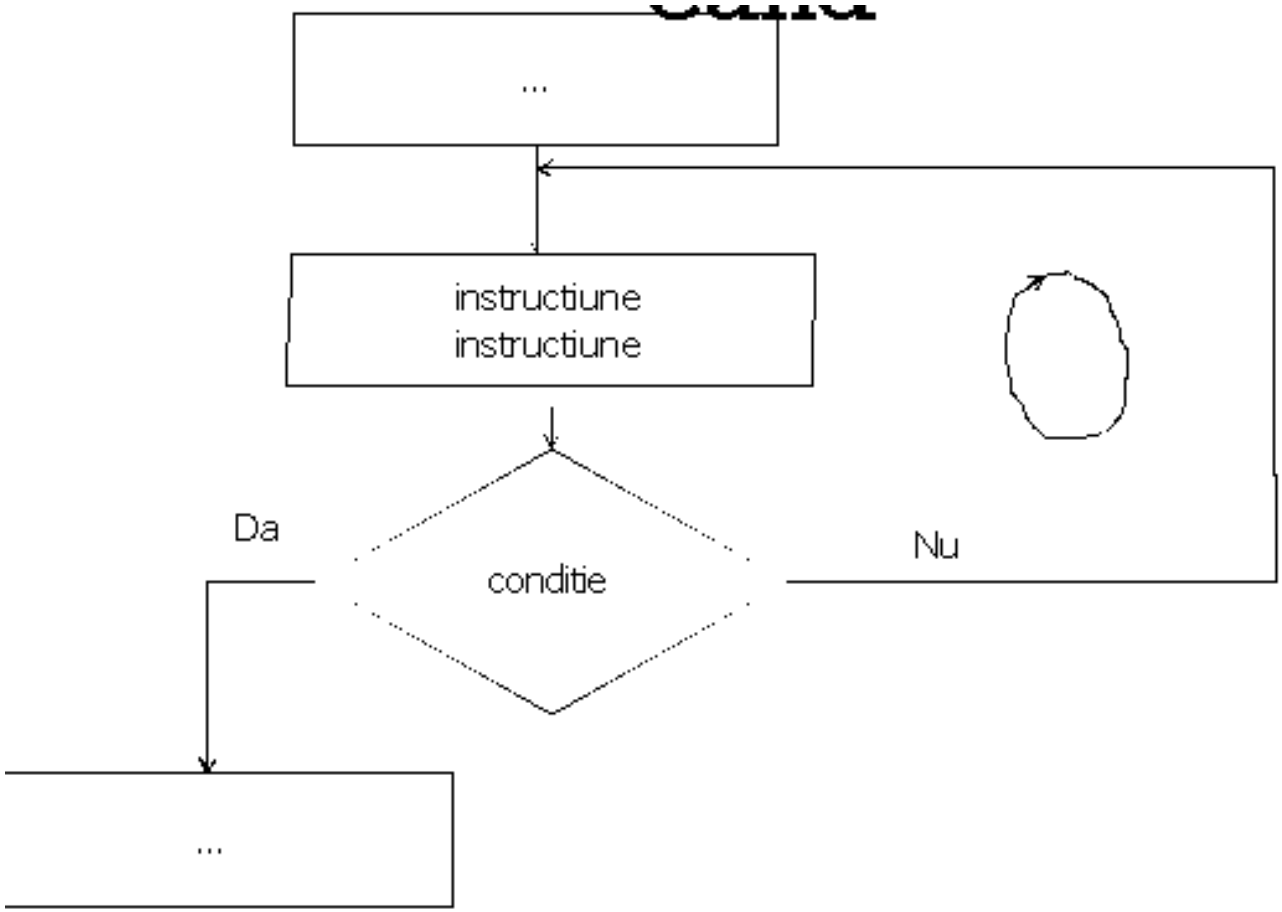
- ***REPETĂ*** *A* ***PÂNĂ CÂND*** *cond* ***SFREP***

- ***Structură echivalentă:***
- ***CÂTTIMP*** *not(cond)* ***EXECUTĂ*** *A* ***SFCÂT***

```
întreg numar;
repetă
    citire numar;
până când numar=0;
```

11.3. repetă ... Până când

- ***REPETĂ*** *A* ***PÂNĂ CÂND*** *cond* ***SFREP***



12. Algoritmi elementari pentru laborator

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. Algoritm pentru interschimbare a două numere
6. Algoritm pentru interschimbare a două numere fără variabilă intermediară
7. Determinarea maximului (minimului) dintr-un șir de numere introduse de la tastatură
8. Extragerea cifrelor unui număr
9. Compunerea unui număr din cifrele sale
10. Determinarea inversului unui număr
11. Algoritmul lui Euclid (prin scădere repetată)
12. Testare număr prim
13. Conversii între sisteme de numerotație