

ALGEBRĂ

Prof. univ. dr. ION D. ION

I. RELAȚII FUNCȚIONALE, RELAȚII DE ECHIVALENTĂ

I.1. Aplicații de mulțimi

Presupunem că cititorul este familiarizat cu teoria naivă a mulțimilor. În spiritul acestei teorii o mulțime este o colecție de obiecte bine determinate, numite **elementele** acesteia. Vom nota mulțimile cu litere majuscule A, B, C, \dots . Dacă A este o mulțime și x un element al lui A , scriem $x \in A$ și citim x **apartine** lui A . Spunem că o mulțime A este **inclusă** în mulțimea B , sau că A este **submulțime** a lui B (scriem $A \subseteq B$), dacă oricare ar fi $x \in A$, avem $x \in B$. Spunem că mulțimea A este **egală** cu mulțimea B , și scriem $A = B$, dacă $A \subseteq B$ și $B \subseteq A$.

Dacă A și B sunt două mulțimi, iar $x \in A$ și $y \in B$, notăm $\{x, y\}$ mulțimea ale cărei elemente sunt x și y . Evident $\{x, y\} = \{y, x\}$. Vom nota cu (x, y) **perechea ordonată**, cu prima componentă x și a doua componentă y .

Mulțimea tuturor perechilor ordonate (x, y) cu $x \in A$ și $y \in B$, notată cu $A \times B$,

$$A \times B = \{(x, y) | x \in A, y \in B\}$$

se numește **produsul cartezian** al lui A cu B .

Dacă (x, y) și (x', y') sunt două elemente ale mulțimii $A \times B$, atunci

$$(x, y) = (x', y') \stackrel{\text{def}}{\Leftrightarrow} x = x' \text{ și } y = y'.$$

Definiția 1. Fie A și B două mulțimi. O submulțime F a lui $A \times B$ se numește **relație funcțională** de la A la B dacă îndeplinește următoarele două condiții:

- (1) $\forall x \in A, \exists y \in B$ astfel încât $(x, y) \in F$.
- (2) dacă $(x, y_1) \in F$ și $(x, y_2) \in F$, atunci $y_1 = y_2$.

Cu alte cuvinte, submulțimea F a lui $A \times B$ este relație funcțională de la A la B dacă oricare ar fi $x \in A$, există un unic $y \in B$ astfel încât $(x, y) \in F$.

Dacă F este o relație funcțională de la A la B , atunci tripletul (A, F, B) , notat f , se numește **aplicație** sau **funcție** de la mulțimea A la mulțimea B ; în aceste condiții A se numește **domeniu** de definiție al aplicației f , B se numește **codomenu** aplicației f , iar F se numește **graficul** lui f .

Alături de notația $f = (A, F, B)$, pentru o aplicație f de la A la B se mai folosesc notățiile $f : A \rightarrow B$ sau $A \xrightarrow{f} B$.

Fie $f = (A, F, B)$ o aplicație de la A la B . Cum F este relație funcțională de la A la B , rezultă că pentru orice $x \in A$ există un unic element $y \in B$ astfel încât $(x, y) \in F$; vom spune că y este imaginea lui x prin aplicația f și va fi notat cu $f(x)$. Cu această convenție, graficul F al aplicației f admite descrierea

$$F = \{(x, f(x)) \mid x \in A\}.$$

Vom spune că f este regula de corespondență (funcțională) de la A la B conform căreia la orice element $x \in A$ se asociază un unic element $y = f(x) \in B$.

Când A și B sunt mulțimi de numere, aplicațiile $f : A \rightarrow B$ se numesc încă **funcții numerice**. În cazul când $A \subseteq \mathbb{R}$ și $B \subseteq \mathbb{R}$, raportând planul euclidian la un reper ortogonal xOy , graficul F al lui f poate fi reprezentat de mulțimea punctelor $P(x, f(x))$ din plan de abscisă $x \in A$ și ordonată $f(x)$. Astfel funcțiile $f : \mathbb{R} \rightarrow \mathbb{R}$ și $g : \mathbb{R} \rightarrow \mathbb{R}$ de grafice $F = \{(x, 2x+3) \mid x \in \mathbb{R}\}$, respectiv $G = \{(x, x^2 - 2x - 3) \mid x \in \mathbb{R}\}$ au reprezentările geometrice din figura 1, respectiv figura 2.

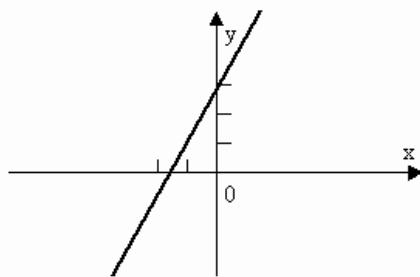


Figura 1
 $y = 2x + 3$

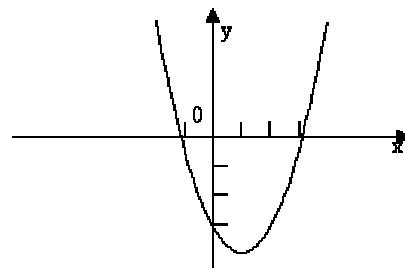


Figura 2
 $y = x^2 - 2x - 3$

Fie $f = (A, F, B)$ și $g = (C, G, D)$ două aplicații de mulțimi. Spunem că aplicația f este **egală** cu aplicația g și scriem $f = g$, dacă și numai dacă $A = C$, $B = D$ și $F = G$. Egalitatea $F = G$ este echivalentă cu $f(x) = g(x)$, $\forall x \in A = C$. Așadar două aplicații de mulțimi $f : A \rightarrow B$ și $g : C \rightarrow D$ sunt egale dacă și numai dacă au același domeniu, același codomeniu și aceeași regulă de corespondență, adică $f(x) = g(x)$, $\forall x \in A = C$.

1.2. Compunerea aplicațiilor de mulțimi

Fie $f : A \rightarrow B$ și $g : B \rightarrow C$ două aplicații de mulțimi astfel încât codomeniul lui f coincide cu domeniul lui g .

Aplicația

$$h : A \rightarrow C, h(x) = g(f(x))$$

se numește compusa lui g cu f (în această ordine!) și se notează cu $g \circ f$.
Așadar

$$g \circ f : A \rightarrow C, (g \circ f)(x) = g(f(x))$$

Din cele de mai sus rezultă că domeniul aplicației $g \circ f$ este A , codomeniul este C iar graficul este

$$\{(x, g(f(x))) \mid x \in A\} \subset A \times C$$

Dacă A este o mulțime, atunci submulțimea Δ_A a lui $A \times A$,

$$\Delta_A = \{(x, x) \mid x \in A\}$$

se numește *diagonala* lui A . Se observă că Δ_A este o corespondență funcțională de la A la A . Aplicația (A, Δ_A, A) se notează cu 1_A și se numește **aplicația identică** a mulțimii A . Cu alte notații, avem:

$$1_A : A \rightarrow A, 1_A(x) = x, \forall x \in A.$$

Teorema 1. *Compunerea aplicațiilor de mulțimi are proprietățile:*

(1) (asociativitate) Oricare ar fi aplicațiile de mulțimi $A \xrightarrow{f} B$, $B \xrightarrow{g} C$ și $C \xrightarrow{h} D$ avem:

$$(h \circ g) \circ f = h \circ (g \circ f)$$

(2) Oricare ar fi aplicația de mulțime $A \xrightarrow{f} B$ avem

$$f \circ 1_A = f \text{ și } 1_B \circ f = f.$$

Fie A o mulțime. Vom nota cu T_A mulțimea tuturor aplicațiilor $f : A \rightarrow A$, numită încă **transformări** ale mulțimii A .

Corolar. Fie A o mulțime. Avem:

(1) $\forall f, g \in T_A \Rightarrow f \circ g \in T_A$

(2) $\forall f, g, h \in T_A \Rightarrow f \circ (g \circ h) = (f \circ g) \circ h$

(3) $\forall f \in T_A \Rightarrow 1_A \circ f = f \circ 1_A = f$.

Definiția 2. Fie $f : A \rightarrow B$ o aplicație de mulțimi. Spunem că aplicația f este:

injectivă dacă oricare ar fi $x_1, x_2 \in A$, $x_1 \neq x_2$ rezultă $f(x_1) \neq f(x_2)$

surjectivă dacă oricare ar fi $y \in B$, există $x \in A$ astfel încât $f(x) = y$

bijectivă dacă este injectivă și surjectivă

Să observăm că o aplicație $f : A \rightarrow B$ este injectivă dacă din $f(x_1) = f(x_2)$ cu $x_1, x_2 \in A$ rezultă întotdeauna $x_1 = x_2$. De asemenea, aplicația $f : A \rightarrow B$ este bijectivă dacă oricare ar fi $y \in B$ există un unic element $x \in A$ astfel încât $f(x) = y$.

Teorema 2. Fie $f : A \rightarrow B$ și $g : B \rightarrow C$ două aplicații de mulțimi și $g \circ f : A \rightarrow C$ compusa acestora.

- (1) Dacă f și g sunt injective (surjective, bijective) atunci $g \circ f$ este aplicație injectivă (respectiv surjectivă, bijectivă)
- (2) Dacă $g \circ f$ este injectivă (surjectivă) atunci f este aplicație injectivă (respectiv g este aplicație surjectivă)

Fie acum $f : A \rightarrow B$ o aplicație bijectivă. Sub această ipoteză, pentru orice $y \in B$ există un unic element $x \in A$ astfel încât $f(x) = y$. Așadar, putem defini aplicația $f^{-1} : B \rightarrow A$ astfel încât pentru orice $y \in B$ avem:

$$f^{-1}(y) = x \Leftrightarrow f(x) = y.$$

Rezultă că

$$f(f^{-1}(y)) = f(x) = y = 1_B(y), \forall y \in B$$

și

$$f^{-1}(f(x)) = x = 1_A(x), \forall x \in A$$

deci

$$f \circ f^{-1} = 1_B \text{ și } f^{-1} \circ f = 1_A$$

Aplicația $f^{-1} : B \rightarrow A$ asociată aplicației bijective $f : A \rightarrow B$ se numește **inversa** lui f .

În particular, dacă $f : A \rightarrow A$ este o aplicație bijectivă, atunci inversa sa satisfacă:

$$f \circ f^{-1} = 1_A = f^{-1} \circ f.$$

I.3. Relații de echivalență. Mulțime factor

Fie A o mulțime nevidă. O submulțime R a lui $A \times A$ se numește **relație binară** pe A . Fie x și y două elemente ale lui A . Spunem că x este în relația R cu y , și scriem xRy , dacă și numai dacă $(x, y) \in R$; dacă $(x, y) \notin R$ spunem că x nu este în relația R cu y , și scriem $x \not R y$.

Spunem că relația binară R este:

reflexivă	dacă $\forall x \in A$ avem xRx
simetrică	dacă din xRy rezultă întotdeauna yRx
tranzitivă	dacă din xRy și yRz rezultă întotdeauna xRz
antisimetrică	dacă din xRy și yRx rezultă $x = y$

Definiția 3. O relație binară R pe mulțimea A reflexivă, simetrică și tranzitivă se numește **relație de echivalență** pe A .

Dacă R este o relație de echivalență pe A , iar $x, y \in A$, atunci în loc de xRy , respectiv $x \not R y$, scriem $x \sim y$, respectiv $x \not\sim y$ și citim x **echivalent cu** y , respectiv x **nu este echivalent cu** y .

Fie A o mulțime nevidă pe care este definită o relație de echivalență " \sim ". Dacă $a \in A$, atunci submulțimea \hat{a} a lui A ,

$$\hat{a} = \{x \in A \mid x \sim a\}$$

se numește **clasa de echivalență** a lui a . Cum $a \sim a$, rezultă că $a \in \hat{a}$, deci \hat{a} este o parte nevidă a lui A .

Să observăm că pentru $a, b \in A$ avem

$$\hat{a} = \hat{b} \Leftrightarrow a \sim b$$

În adevăr, dacă $\hat{a} = \hat{b}$, din $a \in \hat{a}$ rezultă $a \in \hat{b}$, deci $a \sim b$.

Reciproc, presupunem $a \sim b$ și fie $x \in \hat{a}$. Din $x \sim a$ și $a \sim b$ rezultă $x \sim b$, deci $x \in \hat{b}$. Așadar, $\hat{a} \subseteq \hat{b}$. Dar din $a \sim b$ rezultă $b \sim a$, deci avem și $\hat{b} \subseteq \hat{a}$, de unde $\hat{a} = \hat{b}$.

Să mai observăm că dacă $a \not\sim b$, atunci $\hat{a} \cap \hat{b} = \emptyset$, căci altfel, dacă $c \in \hat{a} \cap \hat{b}$, avem $c \sim a$ și $c \sim b$, adică $a \sim b$ (contradicție). Rezultă că două clase de echivalență sau coincid sau sunt disjuncte, ele nu se pot intersecta parțial.

Dacă R este o relație de echivalență pe mulțimea A vom nota cu $\frac{A}{R}$

mulțimea tuturor claselor de echivalență \hat{a} ale elementelor $a \in A$,

$$\frac{A}{R} = \{\hat{a} \mid a \in A\}$$

Mulțimea $\frac{A}{R}$ se numește **mulțimea factor** a lui A prin relația de echivalență R . Când relația de echivalență este notată cu " \sim ", mulțimea factor se notează cu $\frac{A}{\sim}$.

Două elemente a, b din A "produc" aceeași clasă de echivalență dacă și numai dacă $a \sim b$ și "produc" elemente distințe ale mulțimii factor $\frac{A}{R}$ dacă și numai dacă $a \neq b$. Legat de acest fenomen, introducem următoarea noțiune:

Definiția 4. Fie " \sim " o relație de echivalență pe mulțimea A . O familie $T = \{a_i\}_{i \in I}$ de elemente $a_i \in A$ se numește **transversală** sau încă **sistem complet de reprezentanți** pentru relația de echivalență " \sim " dacă sunt satisfăcute condițiile:

- (1) $\forall i, j \in I, i \neq j \Rightarrow a_i \neq a_j$,
- (2) $\forall x \in A, \exists i \in I$ astfel încât $x \sim a_i$.

Dacă $T = \{a_i\}_{i \in I} \subseteq A$ este o transversală pentru relația de echivalență " \sim "

definită pe A , atunci mulțimea factor $\frac{A}{\sim}$ admite descrierea $\frac{A}{\sim} = \{\hat{a}_i | i \in I\}$.

În adevăr, dacă $a \in A$, atunci există $i \in I$ astfel încât $a \sim a_i$. Rezultă $\hat{a} = \hat{a}_i$, de unde $\frac{A}{\sim} \subseteq \{\hat{a}_i | i \in I\}$. Incluziunea contrară este evidentă.

Astfel, dacă R_n este congruența modulo n pe \mathbb{Z} , $T = \{0, 1, 2, \dots, n-1\}$ este o transversală pentru R_n . Rezultă că mulțimea factor $\frac{\mathbb{Z}}{R_n}$, notată de regulă cu \mathbb{Z}_n , admite descrierea $\mathbb{Z}_n = \{\hat{0}, \hat{1}, \hat{2}, \dots, \hat{n-1}\}$.

II. LEGI DE COMPOZIȚIE. MONOIZI

II.1. Operații algebrice binare

Fie M o mulțime nevidă. Numim **lege de compozиie** sau **operatie algebrică (binară)** pe mulțimea M o aplicație $\varphi : M \times M \rightarrow M$. Așadar, o lege de compozиie pe M este o corespondență funcțională φ de la $M \times M$ la M conform căreia la orice pereche ordonată (x, y) de elemente $x, y \in M$ se asociază un unic element, notat $\varphi(x, y)$, din M , numit **compusul** lui x cu y (în această ordine!).

În locul notației funcționale $\varphi(x, y)$ pentru compusul lui x cu y sunt folosite notații mai convenabile, cum sunt:

- Notăția multiplicativă:** $\varphi(x, y) = xy$ sau $\varphi(x, y) = x \cdot y$. În acest caz operația φ se numește **înmulțire**, xy se numește **produsul** lui x cu y , iar x și y se numesc **factorii** produsului.
- Notăția aditivă:** $\varphi(x, y) = x + y$. În acest caz operația φ se numește **adunare**, $x + y$ se numește **suma** lui x cu y , iar x și y se numesc **termenii** sumei.

Alte notații utilizate pentru compusul $\varphi(x, y)$ al lui x cu y sunt: x^*y (notația **star**), $x \overline{|} y$ (notația **true**), $x \perp y$ (notația **antitrue**), $x \circ y$ etc.

Definiția 1. Fie $\varphi : M \times M \rightarrow M$ o lege de compozitie pe mulțimea M .

Spunem că legea de compozitie φ este **asociativă** dacă

$$\forall x, y, z \in M \Rightarrow \varphi(\varphi(x, y), z) = \varphi(x, \varphi(y, z)).$$

Legea de compozitie φ este **comutativă** dacă

$$\forall x, y \in M \Rightarrow \varphi(x, y) = \varphi(y, x).$$

Un element $e \in M$ se numește **element neutru** pentru legea de compozitie φ dacă

$$\forall x \in M \Rightarrow \varphi(e, x) = x = \varphi(x, e).$$

În notația multiplicativă (aditivă) condițiile de asociativitate și comutativitate se scriu

$$(xy)z = x(yz), \quad xy = yx,$$

respectiv

$$(x + y) + z = x + (y + z), \quad x + y = y + x.$$

Să observăm că elementul neutru, în caz că există, este unic. În adevăr, dacă e_1 și e_2 sunt elemente neutre pentru operația φ , avem $e_1 = \varphi(e_1, e_2)$ pentru că e_2 este element neutru pentru φ . De asemenea $\varphi(e_1, e_2) = e_2$ pentru că și e_1 este element neutru pentru φ . Rezultă că $e_1 = e_2$.

Definiția 2. Fie $\varphi : M \times M \rightarrow M$ o lege de compozitie care admite element neutru e . Spunem că un element $x \in M$ este **simetrizabil** în raport cu operația φ dacă există $x' \in M$ astfel încât

$$\varphi(x, x') = \varphi(x', x) = e.$$

În acest caz x' este numit **simetric** al lui x . În notația multiplicativă pentru φ avem $xx' = x'x = e$

Teorema 1. Fie $\varphi: M \times M \rightarrow M$ o lege de compoziție asociativă și cu element neutru e . Avem:

- (1) simetricul unui element $x \in M$, în caz că există, este unic.
- (2) Dacă elementele x și y sunt simetrizabile, atunci $\varphi(x, y)$ este simetrizabil și

$$\varphi(x, y)' = \varphi(y', x'),$$

ceea ce în notația multiplicativă se scrie $(xy)' = y'x'$.

Observație. Dacă pentru legea de compoziție φ folosim notația aditivă, atunci elementul neutru se notează cu **0** și se numește **elementul zero**, iar în notația multiplicativă se mai notează și cu **1** și se numește **elementul unitate**. De asemenea, simetricul x' al unui element x , în notația aditivă se scrie $-x$ și se numește **opusul** lui x , iar în notația multiplicativă se scrie x^{-1} și se numește **inversul** lui x .

Așadar avem $x + (-x) = (-x) + x = 0$, $-(x + y) = (-y) + (-x)$, respectiv $xx^{-1} = x^{-1}x = 1$, $(xy)^{-1} = y^{-1}x^{-1}$

II.2. Legi de compoziție induse

II.2.1. Lege de compoziție indusă pe o parte stabilă

Fie $\varphi: M \times M \rightarrow M$ o lege de compoziție pe mulțimea M și H o submulțime a lui M . Spunem că H este parte stabilă a lui M în raport cu operația φ , dacă $\forall x, y \in H \Rightarrow \varphi(x, y) \in H$.

În aceste condiții putem defini pe H legea de compoziție $\varphi': H \times H \rightarrow H$ astfel $\forall x, y \in H$, $\varphi'(x, y) \stackrel{\text{def}}{=} \varphi(x, y) \in H$ numită legea de compoziție pe H **indusă** de φ .

Teorema 2. Fie $\varphi: M \times M \rightarrow M$ o lege de compoziție pe M , $H \subseteq M$ o parte stabilă a lui M în raport cu φ și φ' legea de compoziție indusă pe H de φ . Avem:

- (1) Dacă φ este asociativă (comutativă) atunci φ' este lege de compoziție asociativă (respectiv comutativă).
- (2) Dacă $e \in M$ este elementul neutru pentru φ și $e \in H$, atunci e este element neutru și pentru φ' .
- (3) Dacă φ este asociativă și cu element neutru astfel încât $e \in H$, atunci un element $x \in H$ este simetrizabil în raport cu φ' dacă și numai dacă este simetrizabil în raport cu φ și simetricul x' al lui x în raport cu φ aparține lui H .

Remarcă. Avem $I_2 \in T_2(\mathbb{R})$. Matricea $A \in T_2(\mathbb{R})$, $A = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix}$ are inversă în $M_2(\mathbb{R})$ dacă și numai dacă $|A| = ac \neq 0$ și $A^{-1} = \frac{1}{ac} \begin{pmatrix} c & -b \\ 0 & a \end{pmatrix}$. Cum $A^{-1} \in T_2(\mathbb{R})$ rezultă că A este inversabilă și în raport cu operația de înmulțire a matricelor din $T_2(\mathbb{R})$ și inversa sa în raport cu această operație este tot A^{-1}

II.2.2. Lege de compoziție indușă pe o mulțime factor

Fie $\varphi : M \times M \rightarrow M$ o lege de compoziție pe mulțimea M . O relație de echivalență R pe mulțimea M se numește **congruentă** în raport cu operația φ dacă, oricare ar fi $x_1, x_2, y_1, y_2 \in M$ astfel încât $x_1 \sim x_2$ și $y_1 \sim y_2$, rezultă întotdeauna $\varphi(x_1, y_1) \sim \varphi(x_2, y_2)$.

În notație aditivă pentru φ aceasta revine la

$$x_1 \sim x_2 \text{ și } y_1 \sim y_2 \Rightarrow x_1 + y_1 \sim x_2 + y_2,$$

iar în notație multiplicativă pentru φ ,

$$x_1 \sim x_2 \text{ și } y_1 \sim y_2 \Rightarrow x_1 y_1 \sim x_2 y_2.$$

Să considerăm din nou cazul general. Așadar, fie $\varphi : M \times M \rightarrow M$ o lege de compoziție pe M , R o congruență pe M în raport cu φ și $\frac{M}{R} = \{\hat{a} | a \in M\}$ mulțimea factor a lui M prin relația de echivalență R .

Pe mulțimea factor $\frac{M}{R}$ putem defini legea de compoziție

$$\hat{\varphi} : \frac{M}{R} \times \frac{M}{R} \rightarrow \frac{M}{R}, \quad \hat{\varphi}(\hat{a}, \hat{b}) = \widehat{\varphi(a, b)}$$

ceea ce în notație multiplicativă (aditivă) pentru φ și $\hat{\varphi}$ revine la $\hat{a}\hat{b} = \widehat{ab}$, respectiv $\hat{a} + \hat{b} = \widehat{a+b}$.

Să observăm că dacă $a_0 \in \hat{a}$ și $b_0 \in \hat{b}$, atunci $a \sim a_0$ și $b \sim b_0$, deci $\varphi(a, b) \sim \varphi(a_0, b_0)$, de unde $\widehat{\varphi(a, b)} = \widehat{\varphi(a_0, b_0)}$. Conchidem că definiția legii de compoziție $\hat{\varphi}$ este corectă, adică $\hat{\varphi}(\hat{a}, \hat{b})$ nu depinde de reprezentanții claselor \hat{a} și \hat{b} folosiți în construcția sa.

Spunem că $\hat{\varphi}$ este **legea de compoziție indușă** de φ pe mulțimea factor $\frac{M}{R}$.

Teorema 3. Fie $\varphi : M \times M \rightarrow M$ o lege de compozitie pe multimea M , R o congruență pe M în raport cu φ și $\hat{\varphi} : \frac{M}{R} \times \frac{M}{R} \rightarrow \frac{M}{R}$ legea de compozitie indușă de φ pe multimea factor $\frac{M}{R}$. Avem

- (1) Dacă φ este asociativă (comutativă) atunci $\hat{\varphi}$ este asociativă (respectiv comutativă).
- (2) Dacă $e \in M$ este element neutru pentru φ , atunci \hat{e} este element neutru pentru $\hat{\varphi}$.
- (3) Dacă $a \in M$ este simetrizabil în raport cu φ , atunci \hat{a} este simetrizabil în raport cu $\hat{\varphi}$ și $\hat{a} = \hat{a}'$, adică simetricul în raport cu $\hat{\varphi}$ al lui \hat{a} este egal cu clasa simetricului lui a în raport cu φ .

Fie $M = \{a_1, a_2, \dots, a_n\}$ o multime finită cu n elemente. Acțiunea unei legi de compozitie φ pe M poate fi descrisă cu ceea ce este cunoscut sub numele de tablă Cayley, care este un tabel cu n linii și n coloane marcate cu elementele mulțimii M . La intersecția liniei lui a_i cu coloana lui a_j în tabla Cayley a operației φ se află $\varphi(a_i, a_j)$, adică compusul lui a_i cu a_j .

φ	a_1	\dots	a_j	\dots	a_j
a_1				\vdots	
\vdots				\vdots	
a_i	\dots	\dots	$\varphi(a_i, a_j)$	\dots	\dots
\vdots				\vdots	
a_n				\vdots	

Astfel tablele Cayley ale adunării și înmulțirii claselor de resturi modulo n definite pe mulțimea finită $\mathbb{Z}_n = \{\hat{0}, \hat{1}, \dots, \hat{n-1}\}$, în cazul $n = 5$ sunt următoarele:

$+$	$\hat{0}$	$\hat{1}$	$\hat{2}$	$\hat{3}$	$\hat{4}$	\bullet	$\hat{0}$	$\hat{1}$	$\hat{2}$	$\hat{3}$	$\hat{4}$
$\hat{0}$	$\hat{0}$	$\hat{1}$	$\hat{2}$	$\hat{3}$	$\hat{4}$	$\hat{0}$	$\hat{0}$	$\hat{0}$	$\hat{0}$	$\hat{0}$	$\hat{0}$
$\hat{1}$	$\hat{1}$	$\hat{2}$	$\hat{3}$	$\hat{4}$	$\hat{0}$	$\hat{1}$	$\hat{0}$	$\hat{1}$	$\hat{2}$	$\hat{3}$	$\hat{4}$
$\hat{2}$	$\hat{2}$	$\hat{3}$	$\hat{4}$	$\hat{0}$	$\hat{1}$	$\hat{2}$	$\hat{0}$	$\hat{2}$	$\hat{4}$	$\hat{1}$	$\hat{3}$
$\hat{3}$	$\hat{3}$	$\hat{4}$	$\hat{0}$	$\hat{1}$	$\hat{2}$	$\hat{3}$	$\hat{0}$	$\hat{3}$	$\hat{1}$	$\hat{4}$	$\hat{2}$
$\hat{4}$	$\hat{4}$	$\hat{0}$	$\hat{1}$	$\hat{2}$	$\hat{3}$	$\hat{4}$	$\hat{0}$	$\hat{4}$	$\hat{3}$	$\hat{2}$	$\hat{1}$

II.3. Monoizi

II.3.1. Definiția monoidului

Într-o primă etapă vom studia obiecte matematice de tipul (M, φ) , unde M este o mulțime nevidă și $\varphi : M \times M \rightarrow M$ o lege de compoziție pe M . O primă clasificare a unor asemenea obiecte matematice se face în funcție de condițiile care se cer să fie satisfăcute de legea de compoziție φ : asociativitate, element neutru, comutativitate etc.

Pe această linie de idei, introducem:

Definiție. Un cuplu (M, φ) format cu o mulțime nevidă M și o lege de compoziție $\varphi : M \times M \rightarrow M$ se numește **monoid** dacă operația φ este asociativă și admite element neutru.

În notație multiplicativă pentru φ , aceasta revine la:

- (1) $\forall x, y, z \in M, (xy)z = x(yz);$
- (2) $\exists e \in M$ astfel încât $ex = xe = x, \forall x \in M.$

Dacă în plus legea de compoziție φ este comutativă, atunci spunem că (M, φ) este monoid comutativ.

Uneori un monoid va fi prezentat ca un triplet (M, φ, e) ; M este **mulțimea suport** a monoidului, $\varphi : M \times M \rightarrow M$ legea de compoziție a monoidului, iar $e \in M$ este elementul neutru. Adesea tripletul (M, φ, e) este notat cu M , la fel cu mulțimea suport.

II.3.2. Produse și sume iterate

Fie (M, \cdot, e) un monoid dat în notația multiplicativă și $x_1, x_2, \dots, x_n \in M$. Vom defini produsul $x_1 x_2 \dots x_n$ al elementelor x_1, x_2, \dots, x_n (în această ordine) după cum urmează: dacă $n = 2$, atunci $x_1 x_2$ este compusul lui x_1 cu x_2 ; dacă $n = 3$, atunci definim $x_1 x_2 x_3 = (x_1 x_2) x_3$; dacă $n = 4$, definim $x_1 x_2 x_3 x_4 = (x_1 x_2 x_3) x_4$, și a.m.d. Așadar produsul $x_1 x_2 \dots x_n$ se definește recurrent prin

$$x_1 x_2 \dots x_n = \begin{cases} x_1 & \text{dacă } n = 1 \\ (x_1 x_2 \dots x_{n-1}) x_n & \text{dacă } n > 1 \end{cases}$$

Deci $x_1 x_2 \dots x_n$ se obține în $n - 1$ pași, prin $n - 1$ înmulțiri, de unde și numele de **produs iterat**.

Se folosește și notația $x_1x_2\dots x_n = \prod_{i=1}^n x_i$ și se citește produs de x_i pentru i de la 1 la n . Cu această convenție avem:

$$\prod_{i=1}^n x_i = \begin{cases} x_1 & \text{dacă } n = 1 \\ \left(\prod_{i=1}^{n-1} x_i \right) x_n & \text{dacă } n > 1 \end{cases}$$

Teorema 4. Fie (M, \cdot, e) un monoid dat în notație multiplicativă. Atunci oricare ar fi $m, n \in \mathbb{N}^*$ și $x_1, x_2, \dots, x_{m+n} \in M$ avem:

$$(x_1x_2\dots x_m)(x_{m+1}x_{m+2}\dots x_{m+n}) = x_1x_2\dots x_{m+n}$$

adică

$$\left(\prod_{i=1}^m x_i \right) \left(\prod_{i=m+1}^{m+n} x_i \right) = \prod_{i=1}^{m+n} x_i .$$

Dacă $x_1 = x_2 = \dots = x_n = a \in M$, atunci produsul $\underbrace{aa\dots a}_{n \text{ factori}}$ se notează cu

a^n (putere a lui a cu exponent număr natural $n > 0$). Cu această convenție de notație, din teorema 4 rezultă:

$$a^m a^n = a^{m+n}, \quad \forall m, n \in \mathbb{N}^*, \quad \forall a \in M$$

Dacă monoidul este dat în notație aditivă $(M, +, 0)$ și $x_1, x_2, \dots, x_n \in M$, atunci definim suma

$$x_1 + x_2 + \dots + x_n = \sum_{i=1}^n x_i ,$$

(se citește sumă de x_i pentru i de la 1 la n) prin:

$$\sum_{i=1}^n x_i = \begin{cases} x_1 & \text{dacă } n = 1 \\ \left(\sum_{i=1}^{n-1} x_i \right) + x_n & \text{dacă } n > 1 \end{cases}$$

Avem:

$$\sum_{i=1}^n x_i + \sum_{i=m+1}^{m+n} x_i = \sum_{i=1}^{m+n} x_i , \quad \forall m, n \in \mathbb{N}^* \text{ și } \forall x_1, x_2, \dots, x_{m+n} \in M .$$

În particular, dacă $x_1 = x_2 = \dots = x_n = a \in M$, suma $\underbrace{a+a+\dots+a}_{n \text{ termeni}}$ se

notează cu na (n multiplu de a) și avem:

$$ma + na = (m+n)a, \quad \forall m, n \in \mathbb{N}^*, \quad \forall a \in M$$

II.3.3. Monoidul liber generat de o mulțime

Fie A o mulțime nevidă numită **alfabet** ale cărei elemente le numim **litere**.

Mulțimea A poate fi de exemplu alfabetul latin. Dacă $n \in \mathbb{N}^*$, definim

$$A^n = \underbrace{A \times A \times \dots \times A}_{n \text{ ori}} = \{(x_1, x_2, \dots, x_n) | x_i \in A, 1 \leq i \leq n\}$$

Așadar, elementele lui A^n sunt sistemele ordonate (x_1, x_2, \dots, x_n) de n elemente din A ; elementele lui A^n se numesc **cuvinte** de lungime n peste alfabetul A .

Fie

$$A^+ = A^1 \cup A^2 \cup \dots \cup A^n \cup \dots = \bigcup_{n=1}^{\infty} A^n.$$

Așadar, A^+ este mulțimea tuturor cuvintelor peste alfabetul A de lungime $1, 2, \dots, n, \dots$

Dacă $u, v \in A^+$, există $m, n \in \mathbb{N}^*$ astfel încât $u \in A^m$ și $v \in A^n$. Avem $u = (x_1, x_2, \dots, x_m)$ și $v = (y_1, y_2, \dots, y_n)$ cu $x_i, y_j \in A$. Prin **juxtapunerea** (**concatenarea**) lui u cu v obținem cuvântul uv ,

$$uv = (x_1, x_2, \dots, x_m, y_1, y_2, \dots, y_n) \in A^{m+n}$$

Se obține astfel o lege de compoziție pe A^+ ,

$$\varphi : A^+ \times A^+ \rightarrow A^+, \varphi(u, v) = uv.$$

Dacă $u, v, w \in A^+$, $u = (x_1, x_2, \dots, x_m)$, $v = (y_1, y_2, \dots, y_n)$, $w = (z_1, z_2, \dots, z_p)$, avem

$$\begin{aligned} (uv)w &= (x_1, \dots, x_m, y_1, \dots, y_n)(z_1, \dots, z_p) = \\ &= (x_1, \dots, x_m, y_1, \dots, y_n, z_1, \dots, z_p) \end{aligned}$$

și

$$\begin{aligned} u(vw) &= (x_1, \dots, x_m)(y_1, \dots, y_n, z_1, \dots, z_p) = \\ &= (x_1, \dots, x_m, y_1, \dots, y_n, z_1, \dots, z_p) \end{aligned}$$

de unde $(uv)w = u(vw)$, deci φ este o lege de compoziție asociativă pe A^+ .

Fie Λ un simbol numit **cuvântul vid** (cuvântul fără litere). Fie $A^* = A^+ \cup \{\Lambda\}$. Prelungim pe A^* înmulțirea de pe A^+ prin

$$\Lambda\Lambda = \Lambda \text{ și } \Lambda u = u\Lambda = u, \forall u \in A^+.$$

Se obține pe A^* o lege de compoziție asociativă având pe Λ ca element neutru. Așadar (A^*, \cdot, Λ) este monoid, numit **monoidul liber** generat de Λ (peste alfabetul A).

Convenim ca să folosim pentru un cuvânt de lungime 1, aceeași notație ca și pentru litera care îl definește. Așadar, identificăm $(x) \equiv x$ pentru orice $x \in A$. Cu această identificare avem $A \subset A^*$.

Să observăm că avem:

$$\begin{aligned} (x_1, x_2) &= (x_1)(x_2) = x_1 x_2 \\ (x_1, x_2, x_3) &= (x_1, x_2)(x_3) = (x_1 x_2)x_3 = x_1 x_2 x_3 \\ &\vdots \\ (x_1, x_2, \dots, x_n) &= (x_1, x_2, \dots, x_{n-1})(x_n) = (x_1 x_2 \dots x_{n-1})x_n = x_1 x_2 \dots x_n \end{aligned}$$

Așadar cuvintele de lungime n peste alfabetul A pot fi scrise ca sevențe $x_1 x_2 \dots x_n$ de n litere din A , $n \in N^*$. Două cuvinte $x_1 x_2, \dots, x_m$ și $y_1 y_2, \dots, y_n$ peste alfabetul A sunt egale dacă și numai dacă $m = n$, adică au aceeași lungime și $x_i = y_i$, $i = 1, \dots, n$, adică au aceleași litere pe poziții identice.

II.3.4. Submonoid. Monoid factor

Fie $M = (M, \varphi, e)$ un monoid. O submulțime N a lui M se numește submonoid al monoidului M dacă sunt îndeplinite condițiile:

- (1) $\forall x, y \in N \Rightarrow \varphi(x, y) \in N$;
- (2) $e \in N$.

Din (1) rezultă că N este parte stabilă a lui M în raport cu φ . Dacă $\varphi' : N \times N \rightarrow N$, $\varphi'(x, y) = \varphi(x, y)$ este legea de compoziție indușă de φ pe N , atunci (N, φ', e) este monoid după cum rezultă din teorema 2, paragraful II.2. Așadar orice submonoid este monoid în raport cu operația indușă.

Fie (M, φ, e) un monoid și R o congruență pe M în raport cu φ . Așadar R este o relație de echivalență pe M cu proprietatea că oricare ar fi $x_1, x_2, y_1, y_2 \in M$

$$x_1 \sim x_2 \text{ și } y_1 \sim y_2 \Rightarrow \varphi(x_1, y_1) \sim \varphi(x_2, y_2)$$

Fie $\frac{M}{R} = \{\hat{a} | a \in M\}$ mulțimea factor a lui M prin congruența R și

$$\hat{\varphi} : \frac{M}{R} \times \frac{M}{R} \rightarrow \frac{M}{R}, \quad \hat{\varphi}(\hat{a}, \hat{b}) = \varphi(\hat{a}, \hat{b}) \text{ lege de compoziție indușă pe } \frac{M}{R} \text{ de } \varphi.$$

Din teorema 3, paragraful II.2. rezultă că legea de compoziție $\hat{\varphi}$ este asociativă și admite ca element neutru pe \hat{e} . Rezultă că $\left(\frac{M}{R}, \hat{\varphi}, \hat{e}\right)$ este monoid, numit

monoidul factor al lui M prin congruența R . Evident $\frac{M}{R}$ este monoid comutativ dacă M este comutativ.

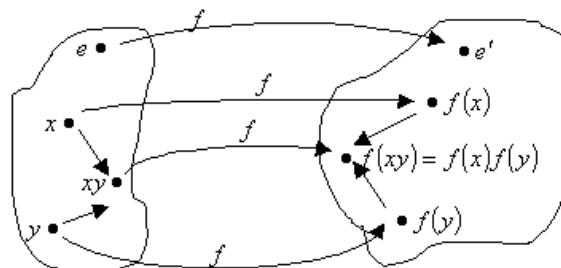
II.3.5. Morfisme de monoizi

Fie (M, \cdot, e) și (M', \cdot, e') doi monoizi. O aplicație $f : M \rightarrow M'$ de la mulțimea suport al primului monoid la mulțimea suport al celui de al doilea monoid se numește **morfism** de monoizi dacă:

- (1) $\forall x, y \in M$ avem $f(xy) = f(x)f(y)$;
- (2) $f(e) = e'$.

Un morfism de monoizi bijectiv se numește **izomorfism** de monoizi.

Așadar un morfism f de la monoidul (M, \cdot, e) la monoidul (M', \cdot, e') este o aplicație f de la mulțimea suport M a primului monoid la mulțimea suport M' a celui de al doilea monoid care aplică pe e peste e' și permute cu operațiile celor doi monoizi: imaginea compusului în M al două elemente $x, y \in M$ este egală cu compusul în M' al lui $f(x)$ și $f(y)$, oricare ar fi $x, y \in M$.



Observație. Dacă monoizii sunt dați în notație aditivă $(M, +, e)$, $(M', +, e')$ atunci condițiile din definiția morfismului de monoizi revin la $f(x+y) = f(x) + f(y)$ și $f(e) = e'$.

III. GRUPURI

III.1. Grup. Reguli de calcul într-un grup

III.1.1. Definiția grupului

Una din cele mai importante structuri ale algebrei este cea de grup. Se numește **grup** un **monoid** G cu toate elementele simetrizabile.

Pentru operația grupului vom folosi de regulă notația multiplicativă. Rezultatele obținute se reformulează ușor și în alte notări pentru operația grupului.

Explicit, noțiunea de grup se introduce astfel:

Definiția 1. O mulțime G nevidă se numește **grup** în raport cu o lege de compoziție internă definită pe G ,

$$G \times G \rightarrow G, (x, y) \mapsto xy$$

dacă sunt îndeplinite condițiile:

- (1) $\forall x, y, z \in G, (xy)z = x(yz)$ (**asociativitate**)
- (2) $\exists e \in G$ astfel încât $ex = xe = x, \forall x \in G$ (**element neutru**)
- (3) $\forall x \in G, \exists x' \in G$ astfel încât $xx' = x'x = e$ (**orice element este simetrizabil**).

Dacă în plus este satisfăcută condiția:

- (4) $\forall x, y \in G, xy = yx$ (**comutativitate**)

atunci G se numește **grup comutativ sau abelian**.

Condițiile (1)-(4) se numesc **axiomele grupului**. Elementul $e \in G$ care satisfac axioma (2) se numește **elementul neutru** al grupului G și este unic determinat. De asemenea, pentru orice $x \in G$, elementul $x' \in G$ cu proprietatea $xx' = x'x = e$ este unic determinat și se numește **simetricul** lui x . Adesea un grup va fi menționat ca un triplet $(G, ;, e)$ format cu mulțimea suport G a grupului G , legea de compoziție $"."$ și elementul neutru e al grupului.

Dacă operația grupului este notată aditiv, atunci elementul neutru se notează cu 0 și se numește **elementul zero** al grupului G . În notație multiplicativă elementul neutru se mai notează cu 1 și se numește **elementul unitate** al grupului G . Simetricul x' al unui element $x \in G$ se notează cu $-x$ și se numește **opusul** lui x (în notație aditivă) și cu x^{-1} și se numește **inversul** lui x (în notație multiplicativă).

III.1.2. Reguli de calcul într-un grup

Teorema 1. Fie $(G, ;, e)$ un grup. Pentru orice $a \in G$, aplicațiile $\lambda_a : G \rightarrow G, \lambda_a(x) = ax$ și $\rho_a : G \rightarrow G, \rho_a(x) = xa$ sunt bijective.

Corolar. Fie $(G, ;, e)$ un grup.

- (1) Dacă $ax = ay$ cu $a, x, y \in G \Rightarrow x = y$ (simplificare la stânga).
- (2) Dacă $xa = ya$ cu $a, x, y \in G \Rightarrow x = y$ (simplificare la dreapta).
- (3) $\forall a, b \in G$ ecuațiile $ax = b$ și $xa = b$ au soluție unică.

Evident, orice grup G este monoid. Dacă $a \in G$ și $n \in \mathbb{N}^*$, definim

$$a^n = \begin{cases} a & \text{dacă } n = 1 \\ a^{n-1}a & \text{dacă } n < 1 \end{cases}$$

și, ca și în cazul monoizilor, avem:

$$a^m a^n = a^{m+n}, \forall m, n \in \mathbb{N}^* \quad (*)$$

Definim $a^0 = e$ și $a^k = (a^{-1})^{-k}$ dacă $k < 0$. Avem:

Teorema 2. Dacă (G, \cdot, e) este grup și $a \in G$, atunci

$$a^h a^k = a^{h+k}, \forall h, k \in \mathbb{Z}$$

Dacă grupul G este dat în notație aditivă, pentru $a \in G$ și $n \in N^*$ definim

$$na = \begin{cases} a & \text{dacă } n = 1 \\ (n-1)a + a & \text{dacă } n > 1 \end{cases}$$

iar pentru $k \in \mathbb{Z}$ $k < 0$, definim $ka = (-k)(-a)$ și avem:

$$(h+k)a = ha + ka, h, k \in \mathbb{Z}.$$

Observație. Dacă grupul G este finit, atunci tabla Cayley a operației grupului este un instrument util în efectuarea calculelor cu elementele grupului.

Pentru $a \in G$, aplicațiile $\lambda_a : G \rightarrow G$, $\lambda_a(x) = ax$ și $\rho_a : G \rightarrow G$, $\rho_a(x) = xa$ sunt bijective. Rezultă că pentru orice $a \in G$ linia (coloana) lui a din tabla Cayley a operației grupului conține fiecare element al grupului o singură dată. Acest fapt se poate observa, de exemplu, pe tabla operației grupului $(\mathbb{Z}_5, +, \hat{0})$ sau a grupului unităților $U(\mathbb{Z}_5) = \{\hat{1}, \hat{2}, \hat{3}, \hat{4}\}$ al monoidului $(\mathbb{Z}_5, \cdot, \hat{1})$.

+	0	1	2	3	4	.	1	2	3	4
0	0	1	2	3	4		1	2	3	4
1	1	2	3	4	0		2	4	1	3
2	2	3	4	0	1		3	1	4	2
3	3	4	0	1	2		4	3	2	1
4	4	0	1	2	3					

Tabla grupului $(\mathbb{Z}_5, +, \hat{0})$

Tabla grupului $U(\mathbb{Z}_5)$

Dacă (G, \cdot, e) este un grup cu trei elemente, $G = \{e, a, b\}$, atunci tabla operației grupului poate fi completată într-un singur fel, anume:

.	e	a	b	.	e	a	b
e	e	a	b	e	e	a	b
a	a	?	?	a	a	b	e
b	b	?	?	b	b	e	a

În adevăr, pe linia (coloana) lui e intră în ordine elementele e, a, b . Pentru a evita repetițiile pe linia lui a , la intersecția cu coloana lui a putem pune b sau e . Dacă punem e , atunci b trebuie pus la intersecția liniei lui a cu coloana lui b , ceea ce produce o repetiție pe coloana lui b . Așadar, la intersecția liniei lui a cu coloana lui a apare b . În continuare pentru a evita repetițiile pe liniile și coloanele tablei Cayley, restul pozițiilor se completează în mod unic. Din tabla operației lui G rezultă acum că $x^3 = e$, $\forall x \in G$. Astfel $a^3 = a^2a = ba = e$.

III.2. Subgrup

III.2.1. Definiția subgrupului

Multe exemple importante de grupuri apar în "interiorul" unor grupuri deja cunoscute, restrângând operația acestora la submulțimi stabilă ale mulțimilor suport. S-a impus astfel conceptul de subgrup al unui grup.

Definiția 1. Fie (G, \cdot, e) un grup. O submulțime nevidă H a lui G se numește subgrup al grupului G dacă

$$(1) \quad \forall x, y \in H \Rightarrow xy \in H;$$

$$(2) \quad \forall x \in H \Rightarrow x^{-1} \in H.$$

Observații.

1. Orice subgrup H conține elementul neutru. În adevăr, cum $H \neq \emptyset$ alegem un element $x \in H$. Avem $x^{-1} \in H$ și deci $e = xx^{-1} \in H$.
2. Orice subgrup H este grup în raport cu operația indușă pe H de operația grupului G , după cum rezultă imediat din teorema 2, capitolul II.
3. Subgrupurile grupului simetric $Sym(A)$ sunt cunoscute sub numele de grupuri de permutări. În secolul al XIX^{lea} conceptul de grup era identic cu cel de grup de permutări. Noțiunea de grup în accepțiunea modernă a fost introdusă de matematicianul englez Arthur Cayley.

III.2.2. Congruența la stânga (dreapta) pe un grup

Fie (G, \cdot, e) un grup. O relație de echivalență " \sim " pe mulțimea suport G a grupului dat se numește **congruență la stânga** dacă pentru orice $x, y \in G$ astfel încât $x \sim y$, rezultă $zx \sim zy$, $\forall z \in G$. Spunem în acest caz că relația " \sim " este **compatibilă la stânga** cu operația lui G .

Dacă H este un subgrup al grupului G și $x, y \in G$, spunem că x este **congruent la stânga** cu y modulo H , și scriem $x \equiv_s y \pmod{H}$, dacă $x^{-1}y \in H$.

Dacă $a \in G$, mulțimea aH ,

$$aH = \{ah \mid h \in H\}$$

se numește **clasă de resturi** la stânga de reprezentant a după subgrupul H .

Să notăm cu R_H^s submulțimea lui $G \times G$ definită prin

$$R_H^s = \{(x, y) \in G \times G \mid x^{-1}y \in H\}.$$

Pentru $x, y \in G$ avem $(x, y) \in R_H^s \Leftrightarrow x \equiv_s y \pmod{H}$.

Teorema 1. Dacă (G, \cdot, e) este un grup și H un subgrup al lui G , atunci relația binară R_H^s este o congruență la stânga pe G . Pentru orice $a \in G$ avem:

$$\hat{a} = \left\{ x \in G \mid x \equiv_s a \pmod{H} \right\} = aH.$$

O relație de echivalență " \sim " pe mulțimea suport G a unui grup (G, \cdot, e) se numește **congruență la dreapta** dacă pentru orice $x, y \in G$ astfel încât $x \sim y$ rezultă $xz \sim yz$, $\forall z \in G$. Vom spune în acest caz că relația " \sim " este **compatibilă la dreapta** cu operația grupului.

Dacă H este un subgrup al grupului G și $x, y \in G$, spunem că x este **congruent la dreapta** cu y modulo H , și scriem $x \equiv_d y \pmod{H}$, dacă $xy^{-1} \in H$. Dacă $a \in G$, mulțimea Ha ,

$$Ha = \left\{ ah \mid h \in H \right\}$$

se numește **clasă de resturi la dreapta** de reprezentant a după subgrupul H . Vom nota cu R_H^d submulțimea lui $G \times G$ definită astfel

$$R_H^d = \left\{ (x, y) \in G \times G \mid xy^{-1} \in H \right\}.$$

Cu o demonstrație asemănătoare celei de la teorema 1, avem:

Teorema 1'. *Dacă (G, \cdot, e) este un grup și H este un subgrup al lui G , atunci relația binară R_H^d este o congruență la dreapta pe G . Pentru orice $a \in G$, avem:*

$$\tilde{a} = \left\{ x \in G \mid x \equiv_d a \pmod{H} \right\} = Ha.$$

Fie $T = \{a_i\}_{i \in I} \subset G$ o transversală pentru relația de echivalență R_H^s pe G . Conform definiției unei transversale (vezi paragraful I.3) avem:

- (1) $\forall i, j \in I, i \neq j \Rightarrow a_i \not\equiv_s a_j \pmod{H}$
- (2) $\forall x \in G, \exists i \in I$ astfel încât $x \equiv_s a_i \pmod{H}$.

Fie $T^{-1} = \{a_i^{-1} \mid a_i \in T\}$. Atunci T^{-1} este o transversală pentru relația de echivalență R_H^d , adică:

- (1*) $\forall i, j \in I, a_i^{-1} \not\equiv_d a_j^{-1} \pmod{H}$
- (2*) $\forall x \in G, \exists i \in I$ astfel încât $x \equiv_d a_i^{-1} \pmod{H}$.

În adevăr, dacă pentru $i \neq j$ avem $a_i^{-1} \equiv_d a_j^{-1} \pmod{H}$, atunci $a_i^{-1}(a_j^{-1})^{-1} \in H$, adică $a_i^{-1}a_j \in H$, ceea ce nu este permis de (1).

Dând $x \in G$, din (2) rezultă că există $a_i \in T$ astfel încât $x^{-1} \equiv_s a_i \pmod{H}$, deci $(x^{-1})^{-1}a_i \in H$, adică $x(a_i^{-1})^{-1} \in H$, de unde $x \equiv_d a_i^{-1} \pmod{H}$.

Cum aplicația $f : T \rightarrow T^{-1}$, $f(a_i) = a_i^{-1}$ este bijectivă, rezultă că și mulțimile factor

$$\frac{G}{R_H^s} = \{aH \mid a \in G\} = \{a_i H \mid a_i \in T\}$$

și

$$\frac{G}{R_H^d} = \{Ha \mid a \in G\} = \{Ha_i^{-1} \mid a_i \in T\}$$

au același cardinal, notat cu $[G : H]$ și numit **indicele** lui H în G .

III.2.3. Teorema lui Lagrange

Fie $(G, ;, e)$ un **grup finit**, adică un grup cu un număr finit de elemente. Fie $n \in \mathbb{N}^*$ numărul elementelor lui G . Numărul n se numește ordinul grupului G și notăm $\text{ord}G = n$. Dacă A este o mulțime putem nota cu $|A|$ numărul elementelor sale. Așadar, dacă G este grup finit, putem scrie $\text{ord}G = |G|$.

Dacă mulțimea suport a lui G este infinită spunem că G este un grup infinit și notăm $\text{ord}G = \infty$.

Teorema 2 (Lagrange). Fie $(G, ;, e)$ un grup finit și H un subgrup al lui G . Atunci

$$|G| = |H| \cdot [G : H].$$

În particular, ordinul oricărui subgrup al unui grup este divizor al ordinului grupului.

III.3. Ordinul unui element. Grupuri ciclice

III.3.1. Definiția ordinului unui element

Fie $(G, ;, e)$ un grup și $a \in G$. Spunem că a este element de **ordin finit** al grupului G dacă există $k \in \mathbb{N}^*$ astfel încât $a^k = e$. Dacă a este element de ordin finit, atunci numărul natural notat cu $\text{ord}(a)$,

$$\text{ord}(a) = \min \{k \in \mathbb{N}^* \mid a^k = e\}$$

se numește **ordinul** lui a . Dacă a nu este element de ordin finit, atunci spunem că a este **element de ordin infinit** și scriem $\text{ord}(a) = \infty$.

Teorema 1. Fie $(G, ;, e)$ un grup.

- (I) Dacă a este un element de ordin $m \in \mathbb{N}^*$ al lui G și $H = \{e, a, a^2, \dots, a^{m-1}\}$, atunci H este subgrup de ordin m al lui G .

- (2) Dacă G este grup finit, atunci orice element $a \in G$ are ordinul finit și $\text{ord}(a) | \text{ord}G$.

Corolar. Fie (G, \cdot, e) un grup finit și $n = |G|$. Atunci $a^n = e$, $\forall a \in G$.

III.3.2. C.m.m.d.c. în \mathbb{Z} . Proprietățile ordinului unui element

Teorema lui Lagrange permite să folosim metode de natură aritmetică în studiul grupurilor finite. Pentru a valorifica această posibilitate să reamintim câteva rezultate din aritmetică numerelor întregi.

Fie $a, b \in \mathbb{Z}$. Un număr întreg $d \geq 0$ se numește **cel mai mare divizor comun** (c.m.m.d.c.) al lui a și b dacă:

- (1) $d | a$ și $d | b$;
- (2) dacă $c | a$ și $c | b$, atunci $c | d$,

Dacă $d' \in \mathbb{Z}$, $d' \geq 0$ satisfacă, de asemenea, (1) și (2), atunci $d | d'$ și $d' | d$, de unde $d = d'$. Așadar, c.m.m.d.c. al numerelor întregi a și b , în caz că există, este unic determinat. Pentru c.m.m.d.c. al lui a și b se folosește notația $d = (a, b)$ (a nu se confunda cu perechea ordonată (a, b) !).

Teorema 2. Pentru orice $a, b \in \mathbb{Z}$ există c.m.m.d.c. al lui a și b . Mai mult, dacă $d = (a, b)$, atunci există $u, v \in \mathbb{Z}$ astfel încât $d = au + bv$.

Dacă $a, b \in \mathbb{Z}$ vom spune că a este **relativ prim** cu b dacă $(a, b) = 1$, ceea ce revine la faptul că există $u, v \in \mathbb{Z}$ astfel încât $au + bv = 1$.

Teorema 3. Fie $a, b, c \in \mathbb{Z}$. Avem:

- (1) dacă $(a, b) = 1$ și $(a, c) = 1$, atunci $(a, bc) = 1$;
- (2) dacă $(a, b) = 1$ și $a | bc$, atunci $a | c$;
- (3) dacă $(a, b) = 1$, $a | c$ și $b | c$, atunci $ab | c$.

Teorema 4. Fie (G, \cdot, e) un grup. Avem:

- (1) Pentru $a \in G$ și $m \in \mathbb{N}^*$ sunt echivalente afirmațiile:
 - (α) $\text{ord}(a) = m$
 - (β) $a^k = e$, $k \in \mathbb{Z} \Leftrightarrow m | k$.
- (2) Fie $a \in G$ astfel încât $\text{ord}(a) = m$. Atunci oricare ar fi $k \in \mathbb{Z}$ avem $\text{ord}(a^k) = \frac{m}{(m, k)}$.

- (3) Dacă $\text{ord}(a) = m$ și $m = dq$, atunci $\text{ord}(a^q) = d$.
(4) Fie $a, b \in G$ astfel încât $\text{ord}(a) = m$, $\text{ord}(b) = n$, $(m, n) = 1$ și $ab = ba$.
Dacă $c = ab$, atunci $\text{ord}(c) = mn$.

III.3.3. Subgrupul generat de o mulțime de elemente ale unui grup. Grupuri ciclice

Fie (G, \cdot, e) și S o mulțime nevidă a lui G . Fie $S^{-1} = \{a^{-1} \mid a \in S\}$ și $X = S \cup S^{-1}$. Dacă $x \in X$, atunci $x^{-1} \in X$. Fie

$$\langle S \rangle = \left\{ x_1 x_2 \dots x_n \mid n \in N^*, x_1, x_2, \dots, x_n \in X = S \cup S^{-1} \right\}.$$

Așadar, $\langle S \rangle$ este mulțimea tuturor produselor finite de elemente din $X = S \cup S^{-1}$. Dacă S este finită, $S = \{a_1, a_2, \dots, a_m\}$, atunci în loc de $\langle \{a_1, a_2, \dots, a_n\} \rangle$ folosim notația $\langle a_1, a_2, \dots, a_n \rangle$. În particular, dacă S are un singur element, $S = \{a\}$, atunci $\langle \{a\} \rangle$ se notează cu $\langle a \rangle$.

Teorema 5. Fie S o submulțime nevidă a grupului (G, \cdot, e) . Atunci:

- (1) $\langle S \rangle$ este un subgrup al lui G și $S \subseteq \langle S \rangle$
(2) Dacă H este subgrup al lui G astfel încât $S \subseteq H$, atunci $\langle S \rangle \subseteq H$.

Altfel spus, $\langle S \rangle$ este cel mai mic subgrup al lui G care conține pe S .

Definiție. Fie (G, \cdot, e) un grup. O submulțime S a lui G astfel încât $G = \langle S \rangle$ se numește **sistem de generatori** pentru grupul G . Spunem că grupul G este **finit generat** dacă admite un sistem finit de generatori, adică există a_1, a_2, \dots, a_n astfel încât $G = \langle a_1, a_2, \dots, a_n \rangle$.

Spunem că grupul G este **ciclic** dacă există $a \in G$ astfel încât $G = \langle a \rangle$.

Dacă $a \in G$, atunci $\langle a \rangle$ se numește **subgrupul ciclic** generat de a .

Să observăm că $\langle a \rangle = \{a^k \mid k \in Z\}$ pentru că orice element $x \in \langle a \rangle$ este un produs finit ai cărui factori sunt egali cu a sau cu a^{-1} . Astfel $aaa^{-1}aaa^{-1} = a^2$, $a^{-1}aa^{-1}a^{-1}a^{-1}a = a^{-3}$ etc.

Dacă grupul G este dat în notație aditivă și $a \in G$, atunci $\langle a \rangle = \{ka \mid k \in Z\}$.

Teorema 6. Fie (G, \cdot, e) un grup ciclic finit de ordin n și $a \in G$ astfel încât $G = \langle a \rangle$. Atunci:

- (1) $\text{ord}(a) = n$ și $G = \{e, a, a^2, \dots, a^{n-1}\}$;
- (2) oricare ar fi $d \in N^*$ divizor al lui n , există un unic subgrup H al lui G astfel încât $|H| = d$.

III.4. Rezultate fundamentale asupra grupului simetric (S_n, \circ, e)

III.4.1. Descompunerea unei permutări în produs de cicluri disjuncte

Am notat cu S_n grupul simetric al mulțimii $A = \{1, 2, \dots, n\}$. Elementele lui S_n sunt aplicațiile bijective $\sigma : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$.

Dacă $\sigma \in S_n$, atunci σ poate fi descrisă cu ajutorul unui tablou cu două linii

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & i & \dots & n \\ \sigma(1) & \sigma(2) & \dots & \sigma(i) & \dots & \sigma(n) \end{pmatrix}$$

în cea de-a doua linie fiind trecute tot numerele $1, 2, \dots, n$ într-o ordine care depinde de σ . Pentru $\sigma(1)$ avem n posibilități. Apoi, îndată ce $\sigma(1)$ a fost fixat, pentru $\sigma(2)$ rămân $n - 1$ posibilități. După ce se alege $\sigma(2)$, pentru $\sigma(3)$ rămân $n - 2$ posibilități, și aşa mai departe. Rezultă că numărul aplicațiilor bijective $\sigma : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$ este egal cu

$$n(n-1)(n-2)\dots2 \cdot 1 = n!$$

deci S_n este un grup de ordin $n!$.

Definiție 1. Dacă $\sigma \in S_n$, atunci mulțimea A_σ ,

$$A_\sigma = \{i \in A \mid \sigma(i) \neq i\} \subseteq \{1, 2, \dots, n\} = A$$

se numește suportul permutării σ . Spunem că permutările $\sigma, \pi \in S_n$ sunt disjuncte dacă $A_\sigma \cap A_\pi = \emptyset$.

Lema 1

- (1) Dacă $\sigma \in S_n$ și $i \in A_\sigma$, atunci $\sigma(i) \in A_\sigma$.
- (2) Dacă permutările $\sigma, \pi \in S_n$ sunt disjuncte atunci $\sigma \circ \pi = \pi \circ \sigma$.

Acțiunea unei permutări $\sigma \in S_n$ asupra numerelor $1, 2, \dots, n$ poate fi descrisă cu ajutorul unei diagrame D_σ într-un plan.

În acest scop asociem numerelor $1, 2, \dots, n$ într-un plan n puncte distincte. Dacă $\sigma(i) = j$, atunci se trasează o săgeată cu originea în punctul asociat lui i și

cu extremitatea în punctul asociat lui j . Cum σ este aplicație bijectivă, din fiecare punct al diagramei "pleacă" o singură săgeată și "sosește" o singură săgeată.

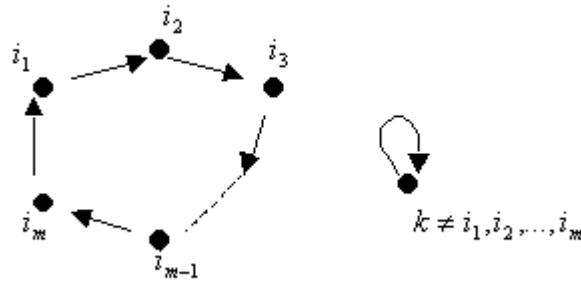
Diagrama $D_{\sigma^{-1}}$ se obține din D_σ inversând sensul săgeților.

Definiția 2. Fie $m \in \mathbb{N}^*$, $m \geq 2$. O permutare $\sigma \in S_n$ se numește **ciclu** de lungime m , sau m -ciclu, dacă există m numere distințe $i_1, i_2, \dots, i_m \in A = \{1, 2, \dots, n\}$ astfel încât $\sigma(i_1) = i_2$, $\sigma(i_2) = i_3$, ..., $\sigma(i_{m-1}) = i_m$, și $\sigma(i_m) = i_1$, oricare ar fi $k \in A \setminus \{i_1, i_2, \dots, i_m\}$. Un 2-ciclu se numește **transpoziție**.

Vom folosi notația $\alpha = (i_1, i_2, \dots, i_m)$ pentru m -ciclul α cu acțiunea pe numerele $1, 2, \dots, n$ descrisă în definiția precedentă. Evident

$$\alpha = (i_1, i_2, \dots, i_m) = (i_2, i_3, \dots, i_m, i_1) = \dots = (i_m, i_1, i_2, \dots, i_{m-1}).$$

Diagrama D_α asociată ciclului α este



Evident, suportul ciclului $\alpha = (i_1, i_2, \dots, i_m)$ este $A_\alpha = \{i_1, i_2, \dots, i_m\}$.

Cum $\alpha^t(i_1) = i_{t+1}$ pentru $t = 1, 2, \dots, m-1$ și $\alpha^m(i) = i$ oricare ar fi $i = 1, 2, \dots, n$, rezultă că $\alpha^t \neq e$, $t = 1, 2, \dots, m-1$ și $\alpha^m = e$. Așadar orice m -ciclu α are ordinul egal cu m . În particular, orice transpoziție $\tau = (i, j)$, $i \neq j$ are ordinul 2, adică $\tau \neq e$ și $\tau^2 = e$.

Teorema 1. Orice permutare $\sigma \in S_n$ se poate reprezenta în mod unic (mai puțin ordinea factorilor) ca produs de cicluri disjuncte.

Lema 2. Fie ciclul $\alpha = (i_1, i_2, \dots, i_m)$. Avem

$$(i_1, i_2, \dots, i_m) = (i_1, i_2) \circ (i_2, i_3) \circ \dots \circ (i_{m-1}, i_m).$$

Teorema 2. Orice permutare $\sigma \in S_n$, $n \geq 2$ se poate reprezenta ca produs finit de transpoziții.

Observație. Dacă $\sigma \in S_n$ este o transpoziție, avem $\sigma \circ \sigma = e$, deci $\sigma^{-1} = \sigma$. Așadar, rezultatul din teorema 2 arată că transpozițiile formează un sistem de generatori pentru grupul S_n .

III.4.2. *Signatura unei permutări*

Dată $\sigma \in S_n$, $n \geq 2$, notăm cu $Inv(\sigma)$ numărul perechilor (i, j) cu $i < j$ astfel încât $\sigma(i) > \sigma(j)$. Vom spune că $Inv(\sigma)$ este numărul **inversiunilor** permutării σ . O permutare σ este **pară (impară)** dacă $Inv(\sigma)$ este număr par (respectiv impar). **Signatura** permutării σ , notată cu $\varepsilon(\sigma)$ sau ε_σ , este prin definiție

$$\varepsilon(\sigma) = (-1)^{Inv(\sigma)} \in \{-1, 1\}.$$

Se observă că o permutare $\sigma \in S_n$ este pară (impară) după cum $\varepsilon(\sigma) = 1$ (respectiv $\varepsilon(\sigma) = -1$)

Descompunerea unei permutări σ în produs de transpoziții nu este unică pentru că dacă $\tau_1, \dots, \tau_m, \tau \in S_n$ sunt transpoziții, atunci:

$$\begin{aligned} \tau_1 \circ \dots \circ \tau_m &= \tau_1 \circ \dots \circ \tau_m \circ \tau \circ \tau = \\ &= \tau_1 \circ \dots \circ \tau_m \circ \tau \circ \tau \circ \tau \circ \tau. \end{aligned}$$

Lema 3. Fie $\sigma \in S_n$ și o transpoziție $\tau = (i, j) \in S_n$. Atunci $\varepsilon(\sigma \circ \tau) = -\varepsilon(\sigma)$, adică $\sigma \circ \tau$ și σ au parități diferite.

Corolar. Transpozițiile unei permutări sunt impare.

Lema 4. Fie $\sigma \in S_n$, $n > 1$ și $\sigma = \tau_1 \circ \tau_2 \circ \dots \circ \tau_m$ o reprezentare a lui σ ca produs de transpoziții. Atunci numerele m și $Inv(\sigma)$ au aceeași paritate și deci $\varepsilon(\sigma) = (-1)^m$.

Teorema 4. Signatura produsului a două permutări este egală cu produsul signaturilor:

$$\varepsilon(\sigma \circ \pi) = \varepsilon(\sigma)\varepsilon(\pi), \quad \forall \sigma, \pi \in S_n.$$

Corolar. Dacă $n > 1$, atunci $A_n = \{\sigma \in S_n | \varepsilon(\sigma) = 1\}$ este un subgrup de ordin $\frac{n!}{2}$ al lui S_n și

$$\sigma \circ \pi \circ \sigma^{-1} \in A_n, \quad \forall \sigma \in S_n \text{ și } \forall \pi \in A_n.$$

III.4.3. Elemente conjugate în grupul S_n

Fie (G, \cdot, e) un grup și $x, y \in G$. Spunem că x este **conjugat** cu y , și scriem $x \sim y$, dacă există $a \in G$ astfel încât $axa^{-1} = y$. Se observă că relația binară " \sim " astfel introdusă este o relație de echivalentă pe G , adică:

$$\begin{aligned} x \sim x, \forall x \in G & \text{ (reflexivitate)} \\ x \sim y \Rightarrow y \sim x & \text{ (simetrie)} \\ x \sim y \text{ și } y \sim z \Rightarrow x \sim z & \text{ (tranzitivitate).} \end{aligned}$$

În adevăr, $exe^{-1} = x$, deci $x \sim x$ pentru orice $x \in G$. Dacă $x \sim y$, avem $axa^{-1} = y$ cu $a \in G$, de unde $a^{-1}y(a^{-1})^{-1} = x$. Deci $y \sim x$. În sfârșit, dacă $x \sim y$ și $y \sim z$, avem $axa^{-1} = y$ și $byb^{-1} = z$ cu $a, b \in G$. Rezultă că $(ba)x(ba)^{-1} = z$, deci $x \sim z$.

Relația de conjugare în grupul $(GL_n(\mathbb{C}), \cdot, I_n)$ se numește **relația de asemănare** a matricelor.

Pentru relația de conjugare pe un grup este important să determinăm numărul claselor de elemente conjugate, adică numărul claselor de echivalență ale relației de conjugare, precum și o transversală a acesteia. În cazul grupului $(GL_n(\mathbb{C}), \cdot, I_n)$ răspunsul la această problemă este dată de teoria Jordan care va fi prezentată într-un capitol ulterior al acestei cărți. Să abordăm această problematică pentru grupul S_n .

Dat un număr natural $n \neq 0$. Sirul n_1, \dots, n_l de numere naturale strict pozitive astfel încât $n_1 \leq n_2 \leq \dots \leq n_l$ și $n_1 + n_2 + \dots + n_l = n$ se numește partiție a lui n . Cum

$$1+1+1+1=1+1+2=2+2=1+3=4$$

numărul partițiilor lui 4 este egal cu 5.

Dacă $\sigma \in S_n$, $\sigma \neq e$ putem scrie descompunerea lui σ în produs de cicluri disjuncte:

$$\sigma = (i_1, i_2, \dots, i_m) \circ (j_1, j_2, \dots, j_p) \circ \dots \circ (k_1, k_2, \dots, k_q)$$

astfel încât $1 < m \leq p \leq \dots \leq q$. Sirul $1, 1, \dots, 1, m, p, \dots, q$ unde pe primele $n - (m + p + \dots + q)$ poziții avem numărul 1, se numește partiția asociată permutării σ .

Lema 5. Fie m -ciclul $\alpha = (i_1, i_2, \dots, i_m)$ și $\sigma \in S_n$. Atunci $\sigma \circ \alpha \circ \sigma^{-1} = (\sigma(i_1), \sigma(i_2), \dots, \sigma(i_m))$. În particular rezultă că conjugatul unui m -cicl este tot un m -cicl.

Teorema 5. Fie $\pi, \pi' \in S_n$, $\pi \neq e$, $\pi' \neq e$. Atunci $\pi \sim \pi'$ dacă și numai dacă partițiile asociate lui π și π' coincid.

III.5. Subgrup normal. Grup factor

III.5.1. Subgrupuri normale

Dând un grup $(G, ;, e)$ ne propunem să descriem congruențele pe G în raport cu operația grupului, adică relațiile de echivalență " \sim " pe G cu proprietatea că oricare ar fi $a, b, c, d \in G$ astfel încât $a \sim b$ și $c \sim d$ să rezulte $ac \sim bd$.

Definiție. Fie $(G, ;, e)$ un grup. Un subgrup N al grupului G se numește subgrup normal al lui G dacă

$$\forall a \in G, \forall x \in N \Rightarrow axa^{-1} \in N.$$

Cu notația $N \trianglelefteq G$ se precizează că N este subgrup normal al lui G .

Observație. Dacă $N \trianglelefteq G$, avem $a^{-1}b \in N$ dacă și numai dacă $ab^{-1} \in N$ adică

$$a \equiv_s b \pmod{N} \Leftrightarrow a \equiv_d b \pmod{N}.$$

În adevăr, presupunem că $a^{-1}b = x \in N$. Atunci $b = ax$ și deci $ab^{-1} = a(ax)^{-1} = ax^{-1}a^{-1} \in N$ pentru că $x^{-1} \in N$. Reciproc, dacă $ab^{-1} = y \in N$, avem $a = yb$ și deci $a^{-1}b = (yb)^{-1}b = b^{-1}y^{-1}b = (b^{-1})y^{-1}(b^{-1})^{-1} \in N$ pentru că $y^{-1} \in N$.

Având în vedere cele de mai sus, dacă $N \trianglelefteq G$ și $a, b \in G$, vom spune că a este **congruent** cu b modulo N și vom scrie $a \equiv b \pmod{N}$ dacă și numai dacă $ab^{-1} \in N$, ceea ce este echivalent cu $a^{-1}b \in N$.

Teorema 1. Fie $(G, ;, e)$ un grup și $N \trianglelefteq G$. Atunci oricare ar fi $a, b, c, d \in G$,

$$a \equiv b \pmod{N} \text{ și } c \equiv d \pmod{N} \Rightarrow ac \equiv bd \pmod{N}$$

adică congruența modulo N este o congruență pe G în raport cu operația lui G .

Reciproc, dacă " \sim " este o congruență pe G în raport cu operația grupului G și $N = \{e\}$, atunci $N \trianglelefteq G$, iar pentru $a, b \in G$ avem

$$a \sim b \Leftrightarrow a \equiv b \pmod{N}.$$

Un grup $(G, ;, e)$ se numește **simplu** dacă are cel puțin două elemente și nu are subgrupuri normale diferite de $\{e\}$ și G . Orice grup G de ordin p , p număr prim, este simplu pentru că din teorema lui Lagrange rezultă că nu are

subgrupuri diferite de $1 = \{e\}$ și G și deci nu are nici subgrupuri normale diferite de 1 și G .

Teorema 2 (Galois). *Dacă $n \geq 5$, atunci grupul altern A_n este simplu.*

Lema 1. *Dacă $n \geq 3$, grupul altern A_n este generat de ciclurile de ordin 3.*

Lema 2. *Fie $N \trianglelefteq A_n$, $n \geq 5$. Dacă N conține un 3-ciclu α , atunci N conține orice 3-ciclu β și deci $N = A_n$.*

III.5.2. Grup factor

Fie $(G, ;, e)$ un grup și H un subgrup al lui G . Dacă $a \in G$, am notat cu $aH = \{ah \mid h \in H\}$ și cu $Ha = \{ha \mid h \in H\}$; aH se numește **clasă de resturi la stânga** de reprezentant a a lui G după subgrupul H , iar Ha se numește **clasă de resturi la dreapta** de reprezentant a a lui G după subgrupul H . Am arătat că

$$aH = \{x \in G \mid x \equiv_s a \pmod{H}\} = \{x \in G \mid x^{-1}a \in H\},$$

respectiv

$$Ha = \{x \in G \mid x \equiv_d a \pmod{H}\} = \{x \in G \mid xa^{-1} \in H\}$$

Avem următoarea caracterizare a subgrupurilor normale:

Teorema 3. *Fie $(G, ;, e)$ un grup și N un subgrup al lui G . Următoarele afirmații sunt echivalente:*

- (1) $N \trianglelefteq G$;
- (2) $aN = Na$, $\forall a \in G$.

Dacă N este un subgrup normal al grupului G vom nota cu $\frac{G}{N}$ mulțimea

factor a lui G prin relația de congruență modulo N . Elementele lui $\frac{G}{N}$ sunt clasele de echivalență ale relației de congruență modulo N ,

$$\frac{G}{N} = \{\hat{a} \mid a \in G\} = \{aN \mid a \in G\}.$$

Cum relația de congruență modulo subgrupul normal N este o congruență pe G în raport cu operația grupului G , putem considera pe mulțimea factor $\frac{G}{N}$ operația indusă de operația grupului G

$$\frac{G}{N} \times \frac{G}{N} \longrightarrow \frac{G}{N}, (\hat{a}, \hat{b}) \mapsto \hat{a}\hat{b} \stackrel{\text{def}}{=} \widehat{ab}.$$

Teorema 4. Fie $(G, ;, e)$ un grup și $N \trianglelefteq G$. Atunci $\frac{G}{N}$ este grup în raport cu operația $\frac{G}{N} \times \frac{G}{N} \longrightarrow \frac{G}{N}$, $(\hat{a}, \hat{b}) \mapsto \hat{a}\hat{b} = \widehat{ab}$, numit grupul factor al lui G prin subgrupul N . Dacă G este grup abelian, atunci $\frac{G}{N}$ este grup abelian.

Observație. Dacă G este un grup și $N \trianglelefteq G$ astfel încât $[G : N] < \infty$, atunci ordinul grupului factor $\frac{G}{N}$ este egal cu $[G : N]$ adică numărul claselor de resturi distincte după subgrupul N . Când G este grup finit, avem $OrdG = ord \frac{G}{N} \cdot ordN$ după cum rezultă din teorema lui Lagrange.

III.6. Morfisme de grupuri

III.6.1. Izomorfisme de grupuri

În definiția grupului este ignorată natura elementelor mulțimii suport. Ceea ce are efect asupra fizionomiei unui grup este cardinalul mulțimii suport și modul în care operația acționează (cu respectarea axiomelor grupului) asupra elementelor acestuia. Vom spune că două grupuri G și G' sunt de același **tip** dacă există o aplicație bijectivă între mulțimile suport ale celor două grupuri care comută cu operațiile acestora. Mai precis:

Definiția 1. Fie $(G, ;, e)$ și $(G', ;', e')$ două grupuri. O aplicație $f : G \rightarrow G'$ se numește izomorfism dacă $f(xy) = f(x)f(y)$ oricare ar fi $x, y \in G$.

Vom spune că grupul G este izomorf cu grupul G' , și scriem $G \simeq G'$, dacă există un izomorfism $f : G \rightarrow G'$.

Un izomorfism $f : G \rightarrow G$ se numește automorfism al grupului G .

Observație. Dacă $f : G \rightarrow G'$ este un izomorfism de grupuri, iar $f^{-1} : G' \rightarrow G$ este inversa aplicației bijective f , atunci f^{-1} este izomorfism de la grupul $(G', ;', e')$ la grupul $(G, ;, e)$. În adevăr, dacă $x', y' \in G'$, iar $x, y \in G$ astfel încât $x' = f(x)$ și $y' = f(y)$, atunci $x'y' = f(x)f(y) = f(xy)$ și deci

$$f^{-1}(x'y') = xy = f^{-1}(x')f^{-1}(y')$$

Astfel, inversa aplicației bijective $f : \mathbb{R} \rightarrow \mathbb{R}_+^*$, $f(x) = 2^x$ este aplicația $f^{-1} : \mathbb{R}_+^* \rightarrow \mathbb{R}$, $f^{-1}(y) = \log_2 y$ și avem

$$f^{-1}(y_1 y_2) = \log_2(y_1 y_2) = \log_2 y_1 + \log_2 y_2 = f^{-1}(y_1) + f^{-1}(y_2),$$
deci f^{-1} este izomorfism de la grupul $(\mathbb{R}_+^*, \cdot, 1)$ la grupul $(\mathbb{R}, +, 0)$.

III.6.2. Morfisme de grupuri

Renunțând la condiția de bijectivitate din definiția izomorfismului de grupuri se obține noțiunea generală de morfism sau omomorfism de grupuri. Așadar:

Definiția 2. Fie (G, \cdot, e) și (G', \cdot, e') două grupuri. O aplicație $f : G \rightarrow G'$ se numește morfism de la grupul G la grupul G' dacă $f(xy) = f(x)f(y)$ oricare ar fi $x, y \in G$.

Dacă $f : G \rightarrow G'$ este un morfism de grupuri, atunci

$$Ker(f) = \{x \in G \mid f(x) = e'\}$$

și

$$Im(f) = \{f(x) \mid x \in G\} = \{x' \in G' \mid \exists x \in G, f(x) = x'\}$$

se numesc **nucleul**, respectiv **imaginea** lui f .

Teorema 1. Fie (G, \cdot, e) și (G', \cdot, e') două grupuri și $f : G \rightarrow G'$ un morfism de grupuri. Atunci

- (1) $f(e) = e'$ și $f(x^{-1}) = (f(x))^{-1}$, oricare ar fi $x \in G$.
- (2) $Ker(f)$ este subgrup normal al lui G .
- (3) $Im(f)$ este un subgrup al lui G' .
- (4) f este injectiv dacă și numai dacă $Ker(f) = 1 = \{e\}$.

III.6.3. Teorema fundamentală de izomorfism

Fie acum $f : G \rightarrow G'$ un morfism de grupuri și $N = Ker(f)$. Cum N este subgrup normal al lui G putem considera grupul factor $\frac{G}{N}$ și morfismul canonic $\varphi : G \rightarrow \frac{G}{N}$, $\varphi(a) = \hat{a} = aN$.

$$\begin{array}{ccc}
 G & \xrightarrow{f} & G' \\
 \downarrow \varphi & & \nearrow f^* \\
 \frac{G}{\text{Ker}(f)} & &
 \end{array}$$

Fie $\hat{a} = aN \in \frac{G}{\text{Ker}(f)}$. Dacă $x \in \hat{a}$, avem $x = au$ cu $u \in \text{Ker}(f)$ și

atunci

$$f(x) = f(au) = f(a)f(u) = f(a)e' = f(a).$$

Rezultă că f este constant pe $\hat{a} = aN$ și putem defini aplicația:

$$f^* : \frac{G}{\text{Ker}(f)} \rightarrow G', \quad f^*(\hat{a}) = f(a)$$

Pentru orice $\hat{a}, \hat{b} \in \frac{G}{\text{Ker}(f)}$ avem

$$f^*(\hat{a}\hat{b}) = f^*(\widehat{ab}) = f(ab) = f(a)f(b) = f^*(a)f^*(b),$$

deci f^* este morfism de la grupul $\frac{G}{\text{Ker}(f)}$ la grupul G' .

Avem

$$f^*(\hat{a}) = e' \Leftrightarrow f(a) = e' \Leftrightarrow a \in \text{Ker}(f) \Leftrightarrow \hat{a} = \hat{e}.$$

Așadar $\text{Ker}(f^*) = \{\hat{e}\}$ de unde rezultă că f^* este morfism injectiv de grupuri. Cum $\text{Im}(f)$ este subgrup al lui G' și $\text{Im}(f^*) = \text{Im}(f)$, rezultă că f^* este morfism bijectiv de la grupul $\frac{G}{\text{Ker}(f)}$ la grupul $\text{Im}(f)$. Așadar avem:

Teorema 2 (fundamentală de izomorfism). Dacă $f : G \rightarrow G'$ este un morfism de grupuri, atunci

$$\text{Im}(f) \simeq \frac{G}{\text{Ker}(f)}$$

Aplicația $f^* : \frac{G}{\text{Ker}(f)} \rightarrow \text{Im}(f)$, $f^*(\hat{a}) = f(a)$ este numită

izomorfismul canonic.

Teorema 3 (structura grupurilor ciclice). Dacă (G, \cdot, e) este un grup ciclic, atunci G este izomorf cu $(\mathbb{Z}, +, 0)$ sau cu $(\mathbb{Z}_n, +, \hat{0})$.

IV. ACȚIUNI DE GRUPURI. APLICAȚII

IV.1. Acțiune a unui grup pe o mulțime

Definiție. Fie (G, \cdot, e) un grup și M o mulțime nevidă. O aplicație $\varphi : G \times M \rightarrow M$ se numește acțiune a grupului G pe mulțimea M dacă

- (1) $\forall a, b \in G, \forall x \in M, \varphi(a, \varphi(b, x)) = \varphi(ab, x)$
- (2) $\forall x \in M, \varphi(e, x) = x$

Imaginea $\varphi(a, x)$ a cuplului $(a, x) \in G \times M$ prin φ va fi notată de regulă cu $a \cdot x$ sau ax . Așadar avem

- (1) $a(bx) = (ab)x, \forall a, b \in G, \forall x \in M$
- (2) $ex = x, \forall x \in M$.

Dacă pe mulțimea $M \neq \emptyset$ avem o acțiune a grupului G , vom spune că M este o G -mulțime.

Pentru $a, b \in G$ avem

$$(\theta_a \circ \theta_b)(x) = \theta_a(\theta_b(x)) = a(bx) = (ab)x = \theta_{ab}(x), \forall x \in M$$

deci $\theta_a \circ \theta_b = \theta_{ab}$. Cum $1_M = \theta_e = \theta_{aa^{-1}} = \theta_a \circ \theta_{a^{-1}}$ și analog $\theta_{a^{-1}} \circ \theta_a = 1_M$, rezultă că $\theta_a \in \text{Sym}(M), \forall a \in G$.

Teorema 1. Fie (G, \cdot, e) un grup și M o G -mulțime. Aplicația

$$f : G \rightarrow \text{Sym}(M), f(a) = \theta_a$$

este un morfism de la grupul (G, \cdot, e) la grupul $(\text{Sym}(M), \circ, 1_M)$.

Dacă $N = \text{Ker}(f)$, atunci N se numește **nucleul** acțiunii φ a lui G pe M . Dacă $N = 1 = \{e\}$, atunci spunem că acțiunea grupului G pe mulțimea M este **fidelă**.

Dacă $\varphi : G \times G \rightarrow G, \varphi(a, x) \rightarrow ax$ este acțiunea regulată a lui G pe G , atunci $\text{Ker}(f) = 1 = \{e\}$ pentru că dacă $a \in \text{Ker}(f)$, atunci $\theta_a = 1_G$, deci $ax = x, \forall x \in G$, de unde $a = e$.

Dacă $\varphi : G \times G \rightarrow G, \varphi(a, x) = axa^{-1}$ este acțiunea prin conjugare a lui G pe G , atunci nucleul acesteia se notează cu Z_G și se numește centrul grupului G . Avem

$$a \in Z_G \Leftrightarrow \theta_a = 1_G \Leftrightarrow axa^{-1} = x, \forall x \in G \Leftrightarrow ax = xa, \forall x \in G$$

și pentru că $Z_G = \text{Ker}(f)$, avem $Z_G \trianglelefteq G$.

Dacă M este o G -mulțime, atunci introducem pe M relația binară " \sim_G " prin

$$x \sim_G y \Leftrightarrow \exists a \in G \text{ astfel încât } ax = y.$$

Cum $ex = x$, rezultă că $x \sim_G x$, $\forall x \in M$. Dacă $x \sim_G y$ și $a \in G$ este astfel încât $ax = y$, atunci $a^{-1}y = a^{-1}(ax) = (a^{-1}a)x = ex = x$, deci $y \sim_G x$. De asemenea, dacă $x \sim_G y$ și $y \sim_G z$, iar $a, b \in G$ sunt astfel încât $ax = y$ și $by = z$, atunci $(ba)x = b(ax) = by = z$, deci $x \sim_G z$. Rezultă că " \sim_G " este o relație de echivalență pe M .

Clasele de echivalență ale relației " \sim_G " se numesc **orbitele** acțiunii lui G pe M ; dacă $x \in M$, notăm $Orb(x) = \hat{x} = \{ax \mid a \in G\}$.

Dacă M este o G -mulțime și $x \in M$, atunci definim **stabilizatorul** în G sau grupul de izotropie al lui x prin

$$Stab_G(x) \stackrel{\text{def}}{=} \{a \in G \mid ax = x\}$$

și se verifică imediat că $Stab_G(x)$ este subgrup al lui G .

Lema 1. Fie M o G -mulțime și $x, y \in M$ astfel încât $ax = y$ cu $a \in G$. Atunci

$$Stab_G(y) = aStab_G(x)a^{-1}.$$

Lema 2. Dacă M este o G -mulțime și $x \in M$, atunci

$$|Orb(x)| = [G : Stab_G(x)],$$

adică, cardinalul orbitei lui x este egal cu indicele în G al stabilizatorului în G al lui x .

Presupunem că M este o G -mulțime finită și $T = \{x_1, x_2, \dots, x_q\} \subseteq M$ o transversală a relației de echivalență " \sim_G " asociată acțiunii lui G pe M . Avem:

$$M = Orb(x_1) \cup Orb(x_2) \cup \dots \cup Orb(x_q) \text{ (reuniune disjunctă)}$$

Teorema 1 (ecuația claselor). Fie M o G -mulțime finită și $T = \{x_1, x_2, \dots, x_q\} \subseteq M$ o transversală a relației de echivalență " \sim_G " asociată acțiunii lui G pe M . Avem

$$|M| = \sum_{j=1}^q [G : Stab_G(x_j)].$$

Fie acum un grup $(G, ;, e)$ și considerăm acțiunea prin conjugare a lui G pe G . Dacă $x \in G$, atunci $Stab_G(x) = \{a \in G \mid axa^{-1} = x\} = \{a \in G \mid ax = xa\}$. În acest caz subgrupul $Stab_G(x)$ se numește centralizatorul în G al lui x și se notează cu $C_G(x)$. Avem $C_G(x) = G \Leftrightarrow |Orb(x)| = 1 \Leftrightarrow x \in Z_G$, centrul

grupului G . O transversală T a relației de echivalență " \sim_G " asociat acțiunii prin conjugare a lui G pe G se formează selecționând câte un element din fiecare orbită, de unde rezultă că $Z_G \subseteq T$.

Teorema 2 (ecuația claselor de elemente conjugate). Fie (G, \cdot, e) un grup finit și $T \subseteq G$ o transversală a relației " \sim_G " asociată acțiunii prin conjugare a lui G pe G . Avem

$$|G| = |Z_G| + \sum_{x \in T \setminus Z_G} [G : C_G(x)]$$

Teorema 3 (Cauchy). Fie (G, \cdot, e) un grup finit, $n = \text{ord}G$ și p un număr prim divizor al lui n . Atunci există $a \in G$ astfel încât $\text{ord}(a) = p$.

Fie p un număr prim. Un grup (G, \cdot, e) se numește p -grup dacă oricare ar fi $a \in G$ există $e \in \mathbb{N}$ astfel încât $\text{ord}(a) = p^e$. Un grup finit G este p -grup dacă și numai dacă $\text{ord}G = p^m$ cu $m \in \mathbb{N}$. În adevăr, dacă $\text{ord}G = p^m$ și $a \in G$, avem $\text{ord}(a) | p^m$, deci $\text{ord}(a) = p^l$, $0 \leq l \leq m$. Reciproc, presupunem că G este un p -grup finit și fie $n = \text{ord}G$. Dacă n nu este de forma p^m , există un număr prim $q \neq p$ astfel încât q divide pe n . Aplicând teorema lui Cauchy, există $a \in G$ astfel încât $\text{ord}(a) = q \neq p^e$. Contradicție.

O proprietate importantă a p -grupurilor finite este:

Teorema 4. Fie $G \neq 1 = \{e\}$ un p -grup finit. Atunci $Z_G \neq 1$.

IV.2. Calculul numărului orbitelor

IV.2.1. Formula Cauchy-Frobenius

Așa cum se va vedea în continuare, în unele aplicații ale teoriei G -mulțimilor este important să știm să calculăm numărul orbitelor. În acest scop, dacă M este o G -mulțime și $a \in G$, definim

$$\text{Fix}(a) = \{x \in M \mid ax = x\} \subseteq M.$$

Dacă G acționează pe o mulțime finită M , definim caracterul permutare asociat acestei acțiuni prin funcția

$$\chi : G \rightarrow \mathbb{N}, \quad \chi(a) = |\text{Fix}(a)|.$$

Astfel, dacă G este un grup finit și G acționează prin conjugare pe G , caracterul permutare asociat este

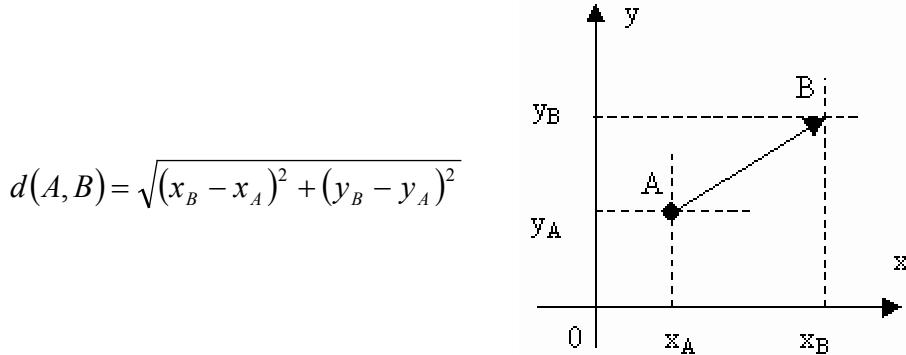
$$\chi : G \rightarrow \mathbb{N}, \quad \chi(a) = |C_G(a)|.$$

Teorema 5 (Cauchy-Frobenius). Dacă grupul finit G acționează pe o mulțime finită M și q este numărul orbitelor, atunci

$$q = \frac{1}{|G|} \sum_{a \in G} \chi(a) = \frac{1}{|G|} \sum_{a \in G} |Fix(a)|$$

IV.2.2. Grupul diedral D_n

Fie \mathbf{P} un plan euclidian. Raportând planul \mathbf{P} la un reper ortonormal xOy , distanța dintre două puncte $A(x_A, y_A)$, $B(x_B, y_B)$ este numărul real $d(A, B)$,



O aplicație $T : \mathbf{P} \rightarrow \mathbf{P}$ se numește *izometrie* (*mișcare rigidă*) dacă conservă distanțele dintre puncte:

$$d(A, B) = d(T(A), T(B)), \forall A, B \in \mathbf{P}$$

Se poate demonstra că orice izometrie este aplicație bijectivă și că două izometrii care coincid pe trei puncte necoliniare coincid pe întreg planul \mathbf{P} ([1]).

Dacă $S : \mathbf{P} \rightarrow \mathbf{P}$ și $T : \mathbf{P} \rightarrow \mathbf{P}$ sunt izometrii, atunci $S \circ T$ este o izometrie. De asemenea, dacă $T : \mathbf{P} \rightarrow \mathbf{P}$ este o izometrie, atunci $T^{-1} : \mathbf{P} \rightarrow \mathbf{P}$ este o izometrie. Așadar, mulțimea $Izom(\mathbf{P})$ a tuturor izometriilor planului \mathbf{P} formează grup în raport cu operația de compunere a aplicațiilor, numit **grupul izometriilor** planului \mathbf{P} .

V. INELE, CORPURI, ALGEBRE

V.1. Inele

V.1.1. Definiția inelului

În această secțiune vom introduce o nouă structură algebrică - structura de inel. Noțiunea de inel are ca prototip mulțimea \mathbb{Z} a numerelor întregi, considerată cu operațiile uzuale de adunare și de înmulțire. Pentru structura de inel există exemple care repezintă mare interes în Algebră (inele de matrice, inele de polinoame), în Analiza matematică (inele de funcții), în Logică (inele booleene).

Definiție. Un triplet $(R, +, \cdot)$, unde R este o mulțime nevidă, iar $"+"$ și $"\cdot"$ sunt două legi de compoziție interne pe R (numite **adunarea** și **înmulțirea**),

$$R \times R \rightarrow R, (x, y) \mapsto x + y,$$

$$R \times R \rightarrow R, (x, y) \mapsto xy,$$

se numește **inel** dacă

(G) $(R, +)$ este grup abelian;

(M) (R, \cdot) este monoid;

(D) înmulțirea este distributivă față de adunare,

$$\forall x, y, z \in R, x(y+z) = xy + xz, (y+z)x = yx + zx.$$

Afirmăția că $(R, +)$ este grup abelian revine la faptul că adunarea unui inel R verifică axioamele:

(G1) $\forall x, y, z \in R, (x+y)+z = x+(y+z);$

(G2) $\exists 0 \in R$ astfel încât $0+x = x+0 = x, \forall x \in R;$

(G3) $\forall x \in R, \exists -x \in R$ astfel încât $x+(-x) = (-x)+x = 0;$

(G4) $\forall x, y \in R, x+y = y+x.$

Afirmăția că (R, \cdot) este morfism revine la faptul că înmulțirea unui inel R este asociativă și admite element neutru:

(M1) $\forall x, y, z \in R, (xy)z = x(yz);$

(M2) $\exists 1 \in R$ astfel încât $1 \cdot x = x \cdot 1 = x, \forall x \in R.$

Vom spune că $(R, +, 0)$ este **grupul aditiv** al inelului R , iar $(R, \cdot, 1)$ este **monoidul multiplicativ** al inelului R . Ansamblul de condiții (G1)–(G4), (M1), (M2), (D) poartă numele de **axiomele inelului**. Elementele 0 și 1 de la axiomele (G2) și (M2) sunt unic determinate (pentru că sunt elemente neutre) și se numesc **elementul zero**, respectiv **elementul unitate** al inelului R . Ele nu sunt obligatoriu numerele reale 0 și 1, natura lor fiind cea a elementelor mulțimii suport R a inelului.

Notă. Un inel se notează de regulă cu R (de la "ring" din limba engleză) sau cu A (de la "anneau" din limba franceză); preferăm prima notație pentru că vom folosi pentru matrice notațiile A, B, \dots . Nu se confundă cu simbolul \mathbb{R} care este utilizat în matematică pentru a nota mulțimea numerelor reale.

Unii autori nu cer în definiția inelului ca înmulțirea să admită element unitate și disting două clase de inele: inele cu element unitate (sau inele unitare) și inele fără element unitate. În această carte prin inel se înțelege inel cu element unitate.

Spunem că un inel R este comutativ dacă înmulțirea este comutativă:

$$(M3) \quad \forall x, y \in R, xy = yx.$$

Dacă $\forall x, y \in R, x \neq 0, y \neq 0$, avem $xy \neq 0$, spunem că R este **inel fără divizori ai lui zero**.

Un inel comutativ R cu $1 \neq 0$ și fără divizori ai lui zero se numește **domeniu de integritate** sau **inel integru**.

V.1.2. Reguli de calcul într-un inel

Fie $(R, +, \cdot)$ un inel. Cum $(R, +)$ este grup abelian, iar (R, \cdot) este monoid, calculul algebric din inelul R beneficiază de regulile de calcul dintr-un grup abelian atunci când este implicată adunarea și de regulile de calcul dintr-un monoid când este implicată înmulțirea. În plus într-un inel avem o serie de reguli de calcul specifice, care angajează ansamblul celor două operații și sunt consecințe ale distributivității înmulțirii față de adunare.

Să observăm că într-un inel R poate fi definită și **operația de scădere** prin

$$R \times R \xrightarrow{\text{def}} R, (x, y) \mapsto x - y = x + (-y)$$

Teorema 1. Dacă $(R, +, \cdot)$ este un inel, atunci

- (1) $\forall x \in R$ avem $x \cdot 0 = 0 \cdot x = 0$;
- (2) dacă $|R| > 1$, atunci $1 \neq 0$;
- (3) (regula semnelor) $x(-y) = (-x)y = -xy$ și $(-x)(-y) = xy$ oricare ar fi $x, y \in R$;
- (4) (distributivitatea înmulțirii față de scădere) $x(y - z) = xy - xz$ și $(y - z)x = yx - zx$ oricare ar fi $x, y, z \in R$;
- (5) dacă R nu are divizori ai lui zero, iar $xy = xz$ sau $yx = zx$ cu $x \neq 0$, atunci $y = z$.

Să mai observăm că într-un inel comutativ sunt adevărate regulile de calcul prescurtat cunoscute pentru numere pentru că stabilirea acestora invocă proprietăți ale adunării și înmulțirii numerelor care se regăsesc și la inele. Mai general, dacă $(R, +, \cdot)$ este un inel, nu neapărat conutativ, iar $a, b \in R$ astfel încât $ab = ba$, atunci

$$\begin{aligned}
(a+b)(a-b) &= a^2 - b^2, \\
(a-b)(a^2 + ab + b^2) &= a^3 - b^3, \\
(a+b)(a^2 - ab + b^2) &= a^3 + b^3, \\
(a+b)^n &= a^n + C_n^1 a^{n-1} b + \dots + C_n^k a^{n-k} b^k + \dots + b^n
\end{aligned}$$

etc.

V.2. Inelul matricelor pătrate

V.2.1. Definiția matricelor

Fie R un inel și $n, m \in \mathbb{N}^*$. Numim **matrice** de tip $m \times n$ cu **coeficienți (intrări)** în inelul R un tablou A format cu mn elemente $a_{ij} \in R$, $1 \leq i \leq m$, $1 \leq j \leq n$,

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mj} & \cdots & a_{mn} \end{pmatrix}$$

dispuse la intersecțiile a m linii și n coloane. Elementele a_{ij} se numesc **coeficienți (intrările)** matricei A . La intersecția liniei i cu coloana j , numită **poziția** (i, j) a matricei A , se află coeficientul a_{ij} ; i este **indicele de linie**, iar j este **indicele de coloană**. Dacă $m = n$, atunci spunem că A este **matrice pătrată** de ordin n ; când $m \neq n$ se spune că A este matrice dreptunghiulară.

Vom nota cu $\mathbf{M}_{m \times n}(R)$ mulțimea tuturor matricelor de tip $m \times n$ cu coeficienți în inelul R și cu $\mathbf{M}_n(R)$ mulțimea tuturor matricelor pătrate de ordin n cu coeficienți în R . Dacă tipul unei matrice A de coeficienți a_{ij} este subîntăles, atunci vom folosi și notația $A = (a_{ij})$.

Fie $A, B \in \mathbf{M}_{m \times n}(R)$, $A = (a_{ij})$, $B = (b_{ij})$. Vom spune că matricea A este egală cu matricea B , și scriem $A = B$, dacă $a_{ij} = b_{ij}$ pentru orice i și j , $1 \leq i \leq m$, $1 \leq j \leq n$. Așadar două matrice A și B sunt declarate egale dacă au același tip și pe poziții egale au aceleași intrări.

V.2.2. Adunarea matricelor

Dacă $A, B \in \mathbf{M}_{m \times n}(R)$, $A = (a_{ij})$, $B = (b_{ij})$, atunci **suma** matricei A cu matricea B , notată cu $A + B$, este matricea $S \in \mathbf{M}_{m \times n}(R)$, $S = (s_{ij})$ astfel încât $s_{ij} = a_{ij} + b_{ij}$, oricare ar fi i și j , $1 \leq i \leq m$, $1 \leq j \leq n$. Așadar, în scriere explicită pentru matrice avem:

$$\begin{aligned} & \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix} = \\ & = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{pmatrix} \end{aligned}$$

Fiind date două matrice A, B de tip $m \times n$, suma lor, notată $A + B$ este o matrice S tot de tip $m \times n$, ai cărei coeficienți se află adunând coeficienții cu aceeași poziție din A și B .

Matricea din $\mathbf{M}_{m \times n}(R)$ cu toți coeficienții egali cu elementul $0 \in R$, notată cu $\mathbf{0}$, se numește matricea zero de tip $m \times n$,

$$\mathbf{0} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

și avem $\mathbf{0} + A = A + \mathbf{0} = A$, oricare ar fi $A \in \mathbf{M}_{m \times n}(R)$.

Dacă $A \in \mathbf{M}_{m \times n}(R)$, matricea $-A$,

$$-A \stackrel{\text{def}}{=} \begin{pmatrix} -a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & -a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{m1} & -a_{m2} & \cdots & -a_{mn} \end{pmatrix},$$

se numește opusa matricei A și avem $A + (-A) = (-A) + A = 0$.

Invocând asociativitatea și comutativitatea adunării inelului R , rezultă că

$$(A + B) + C = A + (B + C)$$

$$A + B = B + A$$

oricare ar fi $A, B, C \in \mathbf{M}_{m \times n}(R)$. Așadar:

Lema 1. Dacă R este un inel, iar $m, n \in \mathbb{N}^*$, atunci $\mathbf{M}_{m \times n}(R)$ este grup abelian în raport cu operația de adunare a matricelor.

V.2.3. Înmulțirea matricelor

În continuare o să definim operația de înmulțire a matricelor. Dacă am folosi modelul de la adunare, am defini produsul a două matrice de tip $m \times n$ să fie matricea de tip $m \times n$ obținută înmulțind coeficienții cu aceeași poziție. Aceasta ar conduce la o operație algebrică lipsită de interes, practic fără aplicații semnificative.

Operația de înmulțire a matricelor, aşa cum va fi definită mai jos, are un corespondent natural în Geometrie (operația de compunere a unor clase de transformări geometrice) și este un instrument major în studiul sistemelor de ecuații liniare.

Pentru ca produsul AB al matricei A cu matricea B (în această ordine) să poată fi efectuat, este necesar ca numărul coloanelor lui A să fie egal cu numărul liniilor lui B (corespondent al condiției ca domeniul să coincidă cu codomeniul în cazul compunerii aplicațiilor). Mai precis dacă A este o matrice de tip $m \times n$, iar B este o matrice de tip $n \times p$, atunci matricea produs $P = AB$ va fi de tip $m \times p$. Pentru orice i și j , $1 \leq i \leq m$, $1 \leq j \leq p$, coeficientul p_{ij} al matricei P este

$$p_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj} = \sum_{k=1}^n a_{ik}b_{kj},$$

adică suma produselor coeficienților liniei i a matricei A cu coeficienții coloanei j a matricei B (pe scurt, produsul liniei i a matricei A cu coloana j a matricei B).

$$i \rightarrow \begin{pmatrix} \vdots & & & \\ a_{i1} & a_{i2} & \cdots & a_{in} \\ \vdots & & & \end{pmatrix} \begin{matrix} \begin{array}{c} j \\ \downarrow \\ \cdots & b_{1j} & \cdots \\ b_{2j} \\ \vdots \\ b_{nj} \end{array} \end{matrix} = \begin{pmatrix} \vdots & & & \\ \cdots & p_{ij} & \cdots \\ \vdots & & \end{pmatrix} \leftarrow i$$

$A \qquad \qquad \qquad B \qquad \qquad \qquad P$

Matricea $I_n \in \mathbf{M}_n(R)$,

$$I_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

care are $1 \in R$ pe diagonală și $0 \in R$ în restul pozițiilor se numește matricea unitate de ordinul n . Din definiția înmulțirii matricilor rezultă:

$$I_m A = A I_n = A, \forall A \in \mathbf{M}_{m \times n}(R)$$

și în particular

$$I_n A = A I_n = A, \forall A \in \mathbf{M}_n(R).$$

Operațiile cu matrice au și proprietățile

$$(AB)C = A(BC) \quad (\text{asociativitate})$$

$$A(B + C) = AB + AC, (B + C)A = BA + CA \quad (\text{distributivitate})$$

cu condiția ca tipurile matricilor A, B, C să fie de așa natură încât operațiile care intervin să fie posibile.

Să observăm că dacă $A, B \in \mathbf{M}_n(R)$, atunci $A + B \in \mathbf{M}_n(R)$ și $AB \in \mathbf{M}_n(R)$, deci adunarea și înmulțirea matricelor sunt legi de compoziție (interne) pe $\mathbf{M}_n(R)$. Acum următorul rezultat este evident.

Teorema 1. *Dacă R este un inel, atunci $\mathbf{M}_n(R)$ este inel în raport cu operațiile de adunare și înmulțire a matricelor.*

Să observăm că dacă $n \geq 2$ și $1 \neq 0$, atunci inelul $\mathbf{M}_n(R)$ nu este comutativ și are divizori ai lui zero. Astfel, dacă $A, B \in \mathbf{M}_2(R)$, $A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \Rightarrow AB = \mathbf{0}, BA = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \neq AB$.

V.2.4. Partiționarea matricelor în blocuri

Data o matrice A de tip $m \times n$, aceasta poate fi descompusă în submatrice disjuncte numite blocuri. Asemenea descompuneri ale matricelor pot simplifica efectuarea calculului matriceal.

Cel mai des vom folosi descompunerea unei matrice A în două blocuri B și C formate cu primele r coloane ale lui A , respectiv ultimele $n - r$ coloane; o asemenea partiționare a lui A se precizează cu ajutorul unei bare verticale trasată între coloanele r și $r + 1$,

$$A = (B|C).$$

Pot fi considerate și alte tipuri de partiționări pentru o matrice A , de exemplu de forma

$$A = \begin{pmatrix} B \\ C \end{pmatrix}, \quad A = \begin{pmatrix} B | C \\ D | E \end{pmatrix}$$

ceea ce se poate scrie mai simplu

$$A = \begin{pmatrix} B & C \end{pmatrix}, \quad A = \begin{pmatrix} B & C \\ D & E \end{pmatrix}.$$

Fie A o matrice de tip $m \times n$, B o matrice de tip $n \times p$, $r \in \mathbb{N}^*$, $1 \leq r < p$ și $B = (C \mid D)$ partiționarea lui B cu blocurile C și D de tip $m \times r$, respectiv $m \times (n-r)$. Au sens produsele AB , AC și AD și aplicând regula de înmulțire a matricelor, avem

$$AB = A(C|D) = (AC|AD)$$

În adevăr, produsul AB se face înmulțind liniile lui A cu coloanele lui B ; înmulțirea cu primele r coloane ale lui B revine la a efectua produsul AC , iar înmulțirea cu ultimele $n-r$ coloane ale lui B revine la a efectua produsul AD .

V.3. Subinele, ideale, inele factor

În acest paragraf vom aborda pentru structura de inel o problematică similară celei de la structura de grup care a impus noțiunile de subgrup, subgrup normal și grup factor. Noțiunile corespunzătoare de la structura de inel vor fi respectiv cele de subinel, ideal, inel factor.

Definiția 1. Fie $(R, +, \cdot)$ un inel. O submulțime S a lui R se numește **subinel** al lui R dacă indeplinește următoarele condiții:

- 1) $\forall x, y \in S \Rightarrow x + y, xy \in S$;
- 2) $\forall x \in S \Rightarrow -x \in S$;
- 3) $1 \in S$.

Așadar, S este subinel al inelului $(R, +, \cdot)$ dacă S este subgrup al grupului $(R, +, 0)$ și submonoid al monoidului $(R, \cdot, 1)$. În particular rezultă că $0 \in S$ pentru că orice subgrup conține elementul neutru al grupului. Să observăm că subinelul S al unui inel R este parte stabilă a lui R în raport cu adunarea și înmulțirea și că S este la rândul său inel în raport cu operațiile induse.

Definiția 2. Fie $(R, +, \cdot)$ un inel. O submulțime nevidă I a lui R se numește **ideal la stânga (dreapta)** a lui R dacă

- (1) $\forall x, y \in I \Rightarrow x + y \in I$;
- (2) $\forall a \in R, \forall x \in I \Rightarrow ax \in I$ (respectiv $xa \in I$).

Un ideal I la stânga și la dreapta al lui R se numește ideal bilateral al lui R și consemnăm aceasta cu notația $I \trianglelefteq R$.

Dacă I este ideal la stânga (dreapta) și $x \in I$, atunci $(-1) \cdot x = -(1 \cdot x) = -x$ și deci $-x = (-1)x \in I$. Rezultă că orice ideal stâng (drept, bilateral) al inelului R este subgrup al grupului $(R, +, 0)$. Dacă inelul R este comutativ, atunci orice ideal al lui R este bilateral.

Fie $(R, +, \cdot)$ un inel și $I \trianglelefteq R$. Așa cum s-a observat, rezultă că I este subgrup al grupului aditiv $(R, +, 0)$ al inelului R . Cum $(R, +, 0)$ este grup abelian, rezultă că I este subgrup normal al lui $(R, +, 0)$ și deci putem considera grupul factor $\left(\frac{R}{I}, +, \hat{0}\right)$ care este abelian. Elementele lui $\frac{R}{I}$ sunt clasele de resturi după subgrupul I ,

$$\frac{R}{I} = \{\hat{a} | a \in R\} = \{a + I | a \in R\},$$

iar operația grupului factor $\frac{R}{I}$ este indușă de adunarea din R ,

$$\hat{a} + \hat{b} \stackrel{\text{def}}{=} \widehat{a+b}.$$

Clasele de resturi $a + I$, $a \in R$ sunt clasele de echivalență ale congruenței modulo I : $a \equiv b \pmod{I} \Leftrightarrow a - b \in I$.

Dar din faptul că I este ideal bilateral al lui R rezultă că congruența modulo I este congruență și în raport cu înmulțirea din R . În adevăr, dacă $a' \equiv a \pmod{I}$ și $b' \equiv b \pmod{I}$, atunci $a' = a + x$ și $b' = b + y$ cu $x, y \in I$.

Avem

$$a'b' = (a+x)(b+y) = ab + ay + xb + xy$$

și cum $ay + xb + xy \in I$, rezultă că $a'b' \equiv ab \pmod{I}$.

Putem considera deci operația indușă pe $\frac{R}{I}$ de înmulțirea de pe R ,

$$\hat{a}\hat{b} \stackrel{\text{def}}{=} \widehat{ab}$$

care este asociativă și admite pe $\hat{1}$ ca element neutru, deci $\left(\frac{R}{I}, \cdot, \hat{1}\right)$ este monoid.

Observând că înmulțirea de pe $\frac{R}{I}$ este distributivă față de adunare, rezultă că

$\left(\frac{R}{I}, +, \cdot\right)$ este inel, numit **inelul factor** al lui R prin idealul bilateral I .

V.4. Morfisme de inele. Teorema fundamentală de izomorfism

Noțiunea de izomorfism de inele permite să evidențiem inele cu aceleși proprietăți algebrice. Ca și la grupuri, vom introduce noțiunea mai generală de morfism între două inele ca fiind aplicațiile între mulțimile lor suport care comută cu operațiile acestora. Mai precis:

Definiție. Fie $(R, +, \cdot)$ și $(R', +, \cdot)$ două inele. O aplicație $f : R \rightarrow R'$ se numește **morfism de inele** dacă oricare ar fi $x, y \in R$ avem

$$f(x+y) = f(x) + f(y), \quad f(xy) = f(x)f(y)$$

și $f(1) = 1'$, unde 1 este elementul unitate al lui R , iar $1'$ cel al lui R' .

Un morfism bijectiv de inele se numește **izomorfism**. Spunem că inelul R este izomorf cu inelul R' și scriem $R \simeq R'$ dacă există cel puțin un izomorfism $f : R \rightarrow R'$.

Să observăm că un morfism de inele $f : R \rightarrow R'$ este în particular morfism de la grupul $(R, +, 0)$ la grupul $(R', +, 0')$ și de la monoidul $(R, \cdot, 1)$ la monoidul $(R', \cdot, 1')$. Rezultă că $f(0) = 0'$, $f(-x) = -f(x)$ oricare ar fi $x \in R$ și pentru orice element inversabil $x \in R$, $f(x)$ este element inversabil al lui R' și $f(x^{-1}) = (f(x))^{-1}$.

Dacă $f : R \rightarrow R'$ este un morfism de inele atunci

$$\text{Ker}(f) = \{x \in R \mid f(x) = 0'\} \text{ și } \text{Im}(f) = \{f(x) \mid x \in R\}$$

se numesc **nucleul**, respectiv **imaginea** lui f .

Teorema 1. Fie $f : R \rightarrow R'$ un morfism de inele.

- (1) $\text{Ker}(f)$ este ideal bilateral al lui R , iar $\text{Im}(f)$ este subinel al lui R' .
- (2) f este injectiv dacă și numai dacă $\text{Ker}(f) = 0$

Teorema 2 (fundamentală de izomorfism). Dacă $f : R \rightarrow R'$ este un morfism de inele, atunci $\text{Im}(f) \simeq \frac{R}{\text{Ker}(f)}$.

V.5. Corp. Corpul fracțiilor unui domeniu

V.5.1. Definiția corpului. Proprietăți

Un cadru ideal pentru efectuarea calculului algebric este dat de inelele în care orice element nenul este inversabil în raport cu înmulțirea.

Definiție 1. Un inel K se numește **corp** dacă $1 \neq 0$ și orice element nenul este simetrizabil în raport cu înmulțirea. Dacă înmulțirea este comutativă, atunci **corpul K** se numește **corp comutativ**.

Observații

1. Orice corp K este inel fără divizori ai lui zero. În adevăr, dacă $ab = 0$ cu $a, b \in K$, iar $a \neq 0$, atunci $b = 1 \cdot b = (a^{-1}a)b = a^{-1}(ab) = a^{-1} \cdot 0 = 0$.
2. Dacă K este un corp și $K^* = K \setminus \{0\}$, atunci K^* este grup în raport cu înmulțirea, numit **grupul multiplicativ** al corpului K .

Teorema 1. *Un domeniu de integritate finit este corp. Inelul $(\mathbb{Z}_p, +, \cdot)$ este corp dacă și numai dacă p este număr prim.*

Notă. O teorema a lui Wedderburn din 1905 stabilește că orice corp finit este comutativ.

Definiție 2. *Fie K și K' două corpuri. O aplicație $f : K \rightarrow K'$ se numește **morfism (izomorfism) de corpuri** dacă este morfism (izomorfism) de la K la K' considerate ca inele. Un morfism (izomorfism) $f : K \rightarrow K$ se numește **endomorfism (respectiv automorfism)** al corpului K .*

Observație. Orice morfism de corpuri este injectiv. În adevăr, fie $f : K \rightarrow K'$ un morfism de corpuri și $x_1, x_2 \in K$ astfel încât $f(x_1) = f(x_2)$.

Arătăm că $x_1 = x_2$. Fie $x = x_1 - x_2$. Avem

$$f(x) = f(x_1 + (-x_2)) = f(x_1) + f(-x_2) = f(x_1) - f(x_2) = 0'$$

Dacă $x \neq 0$, atunci

$$1' = f(1) = f(xx^{-1}) = f(x)f(x^{-1}) = 0'f(x^{-1}) = 0'$$

deci $1' = 0'$. Contradicție.

V.5.2. **Corpul fracțiilor unui domeniu**

Dacă R este un domeniu de integritate, arătăm că există un corp comutativ K astfel încât R să fie subinel al lui K și orice element $x \in R$ poate fi scris sub forma $x = ab^{-1}$ cu $a, b \in R$, $b \neq 0$.

Această descriere a elementelor $x \in R$ cu perechile de elemente (a, b) din R cu $b \neq 0$ nu va fi unică. În adevăr, dacă (c, d) este o altă pereche de elemente din R cu $d \neq 0$ astfel încât $x = ab^{-1} = cd^{-1}$, din egalitatea $ab^{-1} = cd^{-1}$ se obține (prin înmulțire cu bd) $ad = bc$. Așadar perechile (a, b) și (c, d) cu $b \neq 0$, $d \neq 0$ produc același element din K dacă și numai dacă $ad = bc$.

Analiza de mai sus sugerează următoarea construcție pentru K pornind de la domeniul de integritate R .

Fie R un domeniu de integritate, $R^* = R \setminus \{0\}$ și $M = R \times R^*$. Pe M introducem următoarea relație binară " \sim ",

$$(a, b) \sim (c, d) \stackrel{\text{def}}{\Leftrightarrow} ad = bc .$$

Cu ipotezele de mai sus avem:

Teorema 2. Relația binară " \sim " este o relație de echivalență pe M . Mai mult, dacă $(a,b) \sim (a',b')$ și $(c,d) \sim (c',d')$, atunci

$$(ad + bc, bd) \sim (a'd' + b'c', b'd') \text{ și } (ac, bd) \sim (a'c', b'd').$$

Dacă $(a,b) \in R \times R^* = M$, clasa de echivalență $\widehat{(a,b)}$ a perechii (a,b)

se notează cu $\frac{a}{b}$ și se numește **fracție** de elemente din R .

Fie

$$K = \left\{ \widehat{(a,b)} \mid a, b \in R, b \neq 0 \right\} = \left\{ \frac{a}{b} \mid a, b \in R, b \neq 0 \right\}$$

multimea tuturor fracțiilor de elemente din domeniul de integritate R . Să observăm că avem

$$\frac{a}{b} = \frac{ac}{bc}, \quad \forall c \in R^*$$

pentru că $(a,b) \sim (ac,bc)$. Așadar fracțiile pot fi amplificate (simplificate) cu orice element $c \neq 0$ din R .

Pe mulțimea K introducem operațiile de **adunare** și de **înmulțire** a fracțiilor prin

$$\frac{a}{b} + \frac{c}{d} \stackrel{\text{def}}{=} \frac{ad + bc}{bd} \text{ și } \frac{a}{b} \cdot \frac{c}{d} \stackrel{\text{def}}{=} \frac{ac}{bd}$$

Aceste definiții sunt corecte (nu depind de reprezentanții folosiți) pentru că dacă $\frac{a}{b} = \frac{a'}{b'}$ și $\frac{c}{d} = \frac{c'}{d'}$, atunci $(a,b) \sim (a',b')$ și $(c,d) \sim (c',d')$. Conform rezultatelor din teorema 2 avem $(ad + bc, bd) \sim (a'd' + b'c', b'd')$ și $(ac, bd) \sim (a'c', b'd')$, deci

$$\frac{ad + bc}{bd} = \frac{a'd' + b'c'}{b'd'}, \quad \frac{ac}{bd} = \frac{a'c'}{b'd'}.$$

K este numit **corpușul fracțiilor** domeniului de integritate R .

Aplicația $f : R \rightarrow K$, $f(a) = \frac{a}{1}$ este morfism injectiv de inele. Faptul că

f este morfism injectiv de inele de la R la K justifică din punct de vedere al algebrei identificarea $\frac{a}{1} = a$, $\forall a \in R$ (pentru că natura elementelor mulțimii suport a unei structuri algebrice poate fi ignorată) și atunci $R = f(R) = \text{Im}(f)$ care este subinel al lui K .

Teorema 3. Dacă R este un domeniu de integritate există un corp comutativ K , numit **corpul fracțiilor** lui R , astfel încât R este subinel al lui K și pentru orice $x \in K$ există $a, b \in R$, $b \neq 0$ astfel încât $x = ab^{-1}$.

Corpul fracțiilor lui \mathbb{Z} se notează cu \mathbb{Q} și se numește **corpul numerelor rationale**. Avem

$$\mathbb{Q} = \left\{ ab^{-1} \mid a, b \in \mathbb{Z}, b \neq 0 \right\} = \left\{ \frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0 \right\}$$

Un alt corp numeric este corpul \mathbb{R} al numerelor reale a cărei construcție face obiectul Analizei matematice. Îndată ce avem construit corpul \mathbb{R} al numerelor reale, se poate efectua cu mijloacele algebrei, construcția corpului \mathbb{C} al numerelor complexe. Elementele lui \mathbb{C} pot fi introduse ca expresii formale

$$a + bi \text{ cu } a, b \in \mathbb{R}$$

unde i este un simbol, astfel încât

$$\begin{aligned} a + bi &= a' + b'i \stackrel{\text{def}}{\Leftrightarrow} a = a' \text{ și } b = b' \\ (a + bi) + (c + di) &\stackrel{\text{def}}{=} (a + c) + (b + d)i \\ (a + bi) \cdot (c + di) &\stackrel{\text{def}}{=} (ac - bd) + (ad + bc)i. \end{aligned}$$

V.6. R – algebri. Algebra polinoamelor

Definiție 1. Fie R un inel comutativ. Un inel \mathbf{A} , nu neapărat comutativ, se numește R – algebra dacă există o lege de compoziție externă pe \mathbf{A} cu operatori în R , adică o aplicație

$$\varphi : R \times \mathbf{A} \rightarrow \mathbf{A}, \quad \varphi(a, x) \stackrel{\text{not}}{=} ax \in \mathbf{A}$$

astfel încât să fie îndeplinite condițiile :

- (1) $(a+b)x = ax + bx$;
 - (2) $a(x+y) = ax + ay$;
 - (3) $a(bx) = (ab)x$;
 - (4) $1 \cdot x = x$;
 - (5) $(ax)y = x(ay) = a(xy)$,
- oricare ar fi $a, b \in R$ și $x, y \in \mathbf{A}$.

Dacă \mathbf{A} este inel comutativ, atunci \mathbf{A} se numește R – algebra comutativă.

Observație. În structura de R – algebra sunt implicate cinci legi de compoziție: adunarea și înmulțirea inelului comutativ R , adunarea și înmulțirea inelului \mathbf{A} și legea de compoziție externă pe \mathbf{A} cu operatori în R , $R \times \mathbf{A} \rightarrow \mathbf{A}$, $(a, x) \mapsto ax$ numită încă înmulțirea cu scalari (din R) a elementelor din \mathbf{A} .

Definiția 2. Fie \mathbf{A} o R -algebră. Un subinel \mathbf{B} al lui \mathbf{A} se numește R -subalgebră a lui \mathbf{A} dacă

$$\forall a \in R, \forall x \in \mathbf{B} \Rightarrow ax \in \mathbf{B}.$$

Evident, orice R -subalgebră este R -algebră în raport de operațiile induse pe \mathbf{B} de adunarea și înmulțirea din inelul \mathbf{A} și de înmulțirea cu scalari.

Fie acum \mathbf{A} o R -algebră și $\alpha \in \mathbf{A}$. Elementele lui \mathbf{A} de forma

$$p = a_0 + a_1\alpha + a_2\alpha^2 + \dots + a_n\alpha^n \text{ cu } n \in \mathbb{N} \text{ și } a_0, a_1, \dots, a_n \in R$$

se numesc **polinoame** în α cu coeficienți în R ; a_i se numește **coeficient de rang i** . Notăm cu $R[\alpha]$ mulțimea tuturor polinoamelor în α cu coeficienți în R . Avem:

Teorema 1. Fie \mathbf{A} o R -algebră și $\alpha \in \mathbf{A}$. Atunci:

(1) $R[\alpha]$ este o R -subalgebră a lui \mathbf{A} și $\alpha \in R[\alpha]$.

(2) Dacă \mathbf{B} este o R -subalgebră a lui \mathbf{A} astfel încât $\alpha \in \mathbf{B}$, atunci $R[\alpha] \subseteq \mathbf{B}$

Teorema precedentă stabilește că $R[\alpha]$ este cea mai mică R -subalgebră a lui \mathbf{A} care conține pe α ; $R[\alpha]$ se numește R -subalgebra lui \mathbf{A} generată de α .

Să considerăm acum în R -algebra seriilor formale \mathbf{A} elementul

$$X = (0,1,0,0,\dots).$$

Conform regulei de înmulțire din această R -algebră, avem

$$X^2 = X \cdot X = (0,0,1,0,\dots)$$

$$X^3 = X^2 \cdot X = (0,0,0,1,0,\dots)$$

:

$$X^n = X^{n-1} \cdot X = (\underbrace{0,0,\dots,0}_{n \text{ ori}}, 1, 0, \dots)$$

:

Fie $R[X]$ R -subalgebra generată de X a R -algebrelor \mathbf{A} a seriilor formale. Elementele lui $R[X]$ sunt polinoamele f în X cu coeficienți în R ,

$$f = a_0 + a_1X + a_2X^2 + \dots + a_nX^n$$

cu $n \in \mathbb{N}$ și $a_0, a_1, \dots, a_n \in R$. Termenul a_0 trebuie interpretat ca fiind produsul dintre scalarul $a_0 \in R$ și elementul unitate $(1,0,0,\dots)$ al lui \mathbf{A} .

Avem

$$\begin{aligned} f &= a_0(1,0,0,\dots) + a_1(0,1,0,\dots) + \dots + a_n(\underbrace{0,\dots,0}_{n \text{ ori}}, 1, 0, \dots) = \\ &= (a_0, 0, 0, \dots) + (0, a_1, 0, \dots) + \dots + (0, \dots, 0, a_n, 0, \dots) = \\ &= (a_0, a_1, \dots, a_n, 0, \dots) \end{aligned}$$

Dacă avem încă un polinom $g \in R[X]$, $g = b_0 + b_1X + \dots$, atunci $f = g \Leftrightarrow (a_0, a_1, \dots) = (b_0, b_1, \dots) \Leftrightarrow a_i = b_i, i = 0, 1, \dots$

Așadar reprezentarea elementelor R -subalgebrei $R[X]$ ca polinoame în X cu coeficienți în R este unică. În particular un polinom $f \in R[X]$ este egal cu polinomul nul $0 + 0X + 0X^2 + \dots$ dacă și numai dacă toți coeficienții lui f sunt egali cu 0. Subalgebra $R[X]$ va fi numită în continuare R -algebra polinoamelor în nedeterminata X ; R -algebra \mathbf{A} va fi notată cu $R[[X]]$, iar elementele sale (a_0, a_1, \dots) se mai scriu $\sum_{n=0}^{\infty} a_n X^n$, numite serii formale în nedeterminata X cu coeficienți în R .

Dacă $f \in R[X]$, $f = a_0 + a_1X + a_2X^2 + \dots$, $f \neq 0$, atunci numărul $n = \max\{i | a_i \neq 0\}$ se numește gradul lui f , notat $n = \text{grad}(f)$ și putem scrie

$$f = a_0 + a_1X + \dots + a_n X^n, a_n \neq 0.$$

Teorema 2. Dacă R este un domeniu de integritate, atunci $R[X]$ este domeniu de integritate și

$$\text{grad}(fg) = \text{grad}(f) + \text{grad}(g)$$

oricare ar fi $f, g \in R[X], f \neq 0, g \neq 0$.

Dacă R este un domeniu de integritate, atunci corpul fracțiilor lui $R[X]$ se notează cu $R(X)$ și se numește **corpul funcțiilor raționale** în nedeterminata X cu coeficienți în R . Avem

$$R(X) = \left\{ \frac{f}{g} \middle| f, g \in R[X], g \neq 0 \right\}$$

și operațiile acestui corp sunt adunarea și înmulțirea fracțiilor

$$\frac{f_1}{g_1} + \frac{f_2}{g_2} = \frac{f_1g_2 + f_2g_1}{g_1g_2}, \quad \frac{f_1}{g_1} \cdot \frac{f_2}{g_2} = \frac{f_1 \cdot f_2}{g_1 \cdot g_2}.$$

V.7. Evaluarea polinoamelor. Rădăcini ale polinoamelor

Fie R un inel comutativ, \mathbf{A} o R -algebră și $\alpha \in \mathbf{A}$. Dacă $f \in R[X]$, $f = a_0 + a_1X + \dots + a_n X^n$, atunci definim elementul $f(\alpha) \in \mathbf{A}$,

$$f(\alpha) \stackrel{\text{def}}{=} a_0 + a_1\alpha + \dots + a_n \alpha^n \in \mathbf{A},$$

numit valoarea în α a polinomului f . În particular, dacă $\mathbf{A} = R[X]$ și $\alpha = X$, atunci $f(X)$ este chiar polinomul f .

Teorema 1. Fie R un inel comutativ, \mathbf{A} o R -algebră și $\alpha \in \mathbf{A}$. Atunci oricare ar fi $f, g \in R[X]$ și $a \in R$ avem

$$(f + g)(\alpha) = f(\alpha) + g(\alpha), \quad (fg)(\alpha) = f(\alpha)g(\alpha), \quad (af)(\alpha) = af(\alpha).$$

Dacă \mathbf{A} și \mathbf{A}' sunt două R -algebrelle, atunci o aplicație $\varphi: \mathbf{A} \rightarrow \mathbf{A}'$ se numește morfism de R -algebrelle dacă este morfism de inele și în plus $\varphi(ax) = a\varphi(x)$, pricară ar fi $a \in R$ și $x \in \mathbf{A}$.

Teorema 2 (proprietatea de universalitate a algebrei polinoamelor). Fie R un inel cimutativ. Oricare ar fi o R -algebră \mathbf{A} și oricare ar fi $\alpha \in \mathbf{A}$, există un singur morfism de R -algebrelle $\varphi: R[X] \rightarrow \mathbf{A}$ astfel încât $\varphi(X) = \alpha$.

Fie $v_\alpha: R[X] \rightarrow \mathbf{A}$, $v_\alpha(f) = f(\alpha)$, numit **morfismul de evaluare** (în α) a polinoamelor din $R[X]$.

Teorema 3 (a restului). Fie R un inel comutativ, $a \in R$ și $f \in R[x]$ un polinom de grad $n > 0$. Atunci există $q \in R[X]$ și $r \in R$ unic determinați astfel încât

$$f(X) = (X - a)q(X) + r.$$

Mai mult, $r = f(a)$; q se numește câtul, iar r se numește restul împărțirii lui f prin $X - a$.

Observație. Calculele precedente pot fi sistematizate folosind un tabel cu două linii. În prima linie sunt trecuți coeficienții lui f în ordinea $a_n, a_{n-1}, \dots, a_1, a_0$, iar în a doua linie sunt inserați, pe măsură ce sunt determinați cu formulele precedente, coeficienții $c_{n-1}, c_{n-2}, \dots, c_1, c_0$ și r

a_n	a_{n-1}	a_{n-2}	\dots	a_1	a_0	
c_{n-1}	c_{n-2}	c_{n-3}	\dots	c_0	r	

Se observă că $c_{n-1} = a_n$, iar c_{i-1} , $i \neq n$ se află adunând la a_i (care este plasau deasupra lui c_{i-1}) pe c_i (deja determinat) înmulțit cu a , iar $r = a_0 + c_0a$. Această modalitate de calcul este cunoscută sub numele de schema lui Horner.

Fie R un inel comutativ, $f \in R[X]$ și \mathbf{A} o R -algebră. Un element $\alpha \in \mathbf{A}$ se numește **rădăcină** (din \mathbf{A}) a polinomului f dacă $f(\alpha) = 0$.

VI. SPAȚII VECTORIALE DE DIMENSIUNE FINITĂ

VI.1 Spații vectoriale

După cum se va putea observa, orice algebră peste un corp comutativ are, în particular, o structură de spațiu vectorial în sensul următor:

Definiție. Fie K un corp comutativ. Un grup abelian $(V, +)$ se numește **spațiu vectorial** peste corpul K dacă există o lege de compozиție externă pe V cu operatori în K ,

$$K \times V \rightarrow V, (a, x) \mapsto ax$$

astfel încât:

- (1) $(a + b)x = ax + bx$,
- (2) $a(x + y) = ax + ay$,
- (3) $a(bx) = (ab)x$,
- (4) $1 \cdot x = x$,

oricare ar fi $x, y \in V$ și oricare ar fi $a, b \in K$.

Dacă V este un spațiu vectorial peste corpul K , atunci elementele lui V se numesc **vectori**, operația de grup abelian de pe V se numește **adunarea vectorilor**, iar elementul neutru al acesteia, notat provizoriu cu θ , se numește **vectorul zero**. Elementele lui K se numesc **scalarî**, iar legea externă $K \times V \rightarrow V$, $(a, x) \mapsto ax$ se numește **înmulțirea vectorilor cu scări**. Spațiile vectoriale peste corpul K se numesc încă K – spații vectoriale sau K – spații liniare.

Dacă V este un K – spațiu vectorial, iar $x, y \in V$, se definește **diferența** dintre x și y prin

$$x - y \stackrel{\text{def}}{=} x + (-y).$$

Am notat cu θ vectorul zero, adică elementul neutru pentru operația de adunare a vectorilor; vom nota cu 0 elementul zero al corpului K . De regulă vectorul zero se notează tot cu 0 . Într-o primă fază considerăm că este preferabil să avem notări distințe pentru vectorul zero și pentru scalarul zero, pentru a evita unele confuzii.

Teorema 1. Fie V un K – spațiu vectorial. Sunt adevărate proprietățile:

- (1) $0x = a\theta = \theta$, oricare ar fi $a \in K$ și $x \in V$.
- (2) (**regula semnelor**) $(-a)x = a(-x) = -ax$ și $(-a)(-x) = ax$, oricare ar fi $a \in K$ și $x \in V$.
- (3) Dacă $ax = \theta$, atunci $a = 0$ sau $x = \theta$.
- (4) $a(x - y) = ax - ay$

Dacă v_1, v_2, \dots, v_n sunt vectori din K -spațiul vectorial V și $a_1, a_2, \dots, a_n \in K$, atunci spunem că vectorul

$$a_1v_1 + a_2v_2 + \dots + a_nv_n$$

este **combinație liniară** de vectorii v_1, v_2, \dots, v_n ; scalarii a_1, a_2, \dots, a_n se numesc **coeficienții** combinației liniare.

Spunem că vectorii v_1, v_2, \dots, v_n sunt **liniar dependenti**, și scriem $\text{dep}_K(v_1, v_2, \dots, v_n)$, dacă există $a_1, a_2, \dots, a_n \in K$, nu toți nuli, astfel încât

$$a_1v_1 + a_2v_2 + \dots + a_nv_n = \theta.$$

În caz contrar, spunem că vectorii v_1, v_2, \dots, v_n sunt **liniar independenți** și scriem $\text{ind}_K(v_1, v_2, \dots, v_n)$. Evident, vectorii v_1, v_2, \dots, v_n sunt liniar independenți dacă $a_1v_1 + a_2v_2 + \dots + a_nv_n = \theta$ numai în cazul $a_1 = a_2 = \dots = a_n = 0$, adică

$$a_1v_1 + a_2v_2 + \dots + a_nv_n = \theta \Rightarrow a_1 = a_2 = \dots = a_n = 0.$$

Fie V un K -spațiu vectorial și vectorii $v_1, v_2, \dots, v_n, v \in V$. Spunem că vectorul v este **combinație liniară** de vectorii v_1, v_2, \dots, v_n , și scriem $v \text{dep}_K(v_1, v_2, \dots, v_n)$, dacă există $a_1, a_2, \dots, a_n \in K$ astfel încât

$$v = a_1v_1 + a_2v_2 + \dots + a_nv_n = \sum_{i=1}^n a_i v_i$$

Lema 1. Fie V un K -spațiu vectorial și $v_1, v_2, \dots, v_n, v \in V$ astfel încât $\text{ind}_K(v_1, v_2, \dots, v_n)$ și $v \text{dep}_K(v_1, v_2, \dots, v_n, v)$. Atunci $v \text{dep}_K(v_1, v_2, \dots, v_n)$.

Fie din nou V un K -spațiu vectorial. Fie S o submulțime a lui V , nu neapărat finită. Spunem că S este o **mulțime de vectori liniar dependenti**, și scriem $\text{dep}_K S$, dacă există un număr finit de vectori $v_1, v_2, \dots, v_n \in S$ astfel încât $\text{dep}_K(v_1, v_2, \dots, v_n)$; în caz contrar spunem că S este o **mulțime de vectori liniar independenți**, și scriem $\text{ind}_K S$. Așadar, avem $\text{ind}_K S$ dacă și numai dacă orice parte finită a lui S este liniar independentă.

De asemenea, dacă $v \in V$, spunem că vectorul v este liniar dependent de mulțimea S de vectori, și scriem $v \text{dep}_K S$, dacă există un număr finit de vectori $v_1, v_2, \dots, v_n \in S$ astfel încât $v \text{dep}_K(v_1, v_2, \dots, v_n)$, adică putem scrie $v = a_1v_1 + a_2v_2 + \dots + a_nv_n$ cu $a_1, a_2, \dots, a_n \in K$.

Dacă S și T sunt două submulțimi ale lui V , nu neapărat finite și $S \subset T$, atunci avem:

$$\text{dep}_K S \Rightarrow \text{dep}_K T, \text{ind}_K T \Rightarrow \text{ind}_K S, v \text{dep}_K S \Rightarrow v \text{dep}_K T.$$

VI.2. Subspații vectoriale

Definiție. Fie V un K -spațiu vectorial. O submulțime nevidă N a lui V se numește **subspațiu vectorial** al lui V dacă:

- (1) $\forall x, y \in N \Rightarrow x + y \in N$,
- (2) $\forall a \in K, \forall x \in N \Rightarrow ax \in N$.

Observație. Orice subspațiu vectorial N al lui V este spațiu vectorial în raport cu operațiile induse pe N de operațiile lui V . În adevăr, dacă $x \in N$, atunci $-x = (-1)x \in N$, deci N este subgrup al grupului $(V, +)$. Așadar, N este grup abelian în raport cu operația indușă pe N de adunarea vectorilor din V . Evident, operația indușă pe N de înmulțirea cu scalari verifică axioamele (1)-(4) din definiția spațiului vectorial. Cum $N \neq \emptyset$, pentru $x \in N$ avem $\theta = 0x \in N$, deci orice subspațiu vectorial conține vectorul zero.

Fie V un k -spațiu vectorial și $v_1, v_2, \dots, v_n \in V$. Notăm cu $\langle v_1, v_2, \dots, v_n \rangle$ mulțimea tuturor combinațiilor liniare de vectorii v_1, v_2, \dots, v_n ,

$$\langle v_1, v_2, \dots, v_n \rangle = \left\{ x \in V \mid x = \sum_{i=1}^n a_i v_i, a_1, a_2, \dots, a_n \in K \right\}.$$

Teorema 2. Fie V un K -spațiu vectorial, $v_1, v_2, \dots, v_n \in V$ și

$N = \langle v_1, v_2, \dots, v_n \rangle$. Avem:

- (1) N este subspațiu vectorial al lui V și $v_1, v_2, \dots, v_n \in N$,
- (2) Dacă L este subspațiu vectorial al lui V și $v_1, v_2, \dots, v_n \in L$, atunci $N \subseteq L$.

Așadar, N este cel mai mic subspațiu vectorial al lui V care conține vectorii v_1, v_2, \dots, v_n .

Dacă V este un K -spațiu vectorial și $v_1, v_2, \dots, v_n \in V$, atunci $\langle v_1, v_2, \dots, v_n \rangle$ se numește **subspațiul lui V generat** de v_1, v_2, \dots, v_n ; vectorii v_1, v_2, \dots, v_n se nemesc în acest caz **generatori** ai subspațiului $N = \langle v_1, v_2, \dots, v_n \rangle$.

Fie V un spațiu vectorial peste corpul K , iar X și Y două subspații ale lui V . Definim mulțimea de vectori $X + Y$ prin

$$X + Y \stackrel{\text{def}}{=} \left\{ z \in V \mid z = x + y \text{ cu } x \in X, y \in Y \right\},$$

numită **suma** subspațiilor X și Y .

Teorema 3. Fie V un K -spațiu vectorial și X și Y două subspații ale lui V . Atunci:

- (1) $X + Y$ este subspațiu al lui V , $X \subseteq X + Y$, $Y \subseteq X + Y$;
- (2) Dacă L este un subspațiu al lui V astfel încât $X \subseteq L$ și $Y \subseteq L$, atunci $X + Y \subseteq L$.

Așadar $X + Y$ este cel mai mic subspațiu vectorial al lui V care conține pe X și Y .

Lema 2. Fie V un K -spațiu vectorial și X, Y două subspații vectoriale ale lui V . Sunt echivalente afirmațiile:

- (1) $X \cap Y = O$
- (2) $x + y = \theta$ cu $x \in X$, $y \in Y$, atunci $x = y = \theta$
- (3) $x + y = x' + y'$ cu $x, x' \in X$, $y, y' \in Y$, atunci $x = x'$ și $y = y'$.

Dacă subspațiile X și Y ale spațiului vectorial V satisfac una (deci toate) din condițiile de mai sus se spune că suma lui X cu Y este **directă** și în acest caz în loc de notația $X + Y$ folosim notația $X \oplus Y$.

Dacă V este un K -spațiu vectorial, iar X_1, X_2, \dots, X_m sunt subspații ale lui V , definim

$$X_1 + X_2 + \dots + X_m = \{x \in V \mid x = x_1 + x_2 + \dots + x_m, x_i \in X_i, 1 \leq i \leq m\}.$$

Se verifică faptul că $X_1 + X_2 + \dots + X_m$ este subspațiu vectorial al lui V , cel mai mic printre cele care conțin pe X_1, X_2, \dots, X_m ; $X_1 + X_2 + \dots + X_m$ se numește **suma** subspațiilor X_1, X_2, \dots, X_m . De asemenea, rezultatul din lema 2 se extinde astfel: *condițiile următoare sunt echivalente*

- (1) $(X_1 + \dots + X_{j-1}) \cap X_j = O$, $j = 2, \dots, m$
- (2) $x_1 + \dots + x_m = \theta$ cu $x_i \in X_i$, $1 \leq i \leq m$, atunci $x_1 = x_2 = \dots = x_m = \theta$
- (3) $x_1 + \dots + x_m = x'_1 + \dots + x'_m$ cu $x_i, x'_i \in X_i$, $1 \leq i \leq m$, atunci $x_1 = x'_1, \dots, x_m = x'_m$.

În acest caz spunem că suma de subspații $X_1 + X_2 + \dots + X_m$ este **directă** și folosim notația $X_1 \oplus X_2 \oplus \dots \oplus X_m$. Astfel, dacă $\text{ind}_K(v_1, \dots, v_m)$ și $X_i = \langle v_i \rangle = \{av_i \mid a \in K\}$, atunci avem suma directă

$$X_1 \oplus X_2 \oplus \dots \oplus X_m = \langle v_1, v_2, \dots, v_m \rangle.$$

VI.3. Bază și dimensiune

Fie V un spațiu vectorial peste corpul K . O submulțime nevidă S a lui V se numește **sistem de generatori** pentru V dacă oricare ar fi $v \in V$ avem $v \text{ dep}_K S$, adică există un număr finit de vectori $v_1, v_2, \dots, v_n \in S$ astfel încât $v = a_1v_1 + a_2v_2 + \dots + a_nv_n$ cu $a_1, a_2, \dots, a_n \in K$. Un sistem S de generatori liniar independenți se numește **bază** a lui V .

În spațiul K^n considerăm vectorii $e_1 = (1, 0, \dots, 0)$, $e_2 = (0, 1, \dots, 0)$, ..., $e_n = (0, 0, \dots, 1)$. Atunci $B = \{e_1, e_2, \dots, e_n\}$ este o bază pentru K^n numită **baza canonică** sau **baza standard** a lui K^n .

Spunem că un K -spațiu vectorial V este **finit generat**, dacă admite un sistem finit de generatori, ceea ce revine la faptul că există $v_1, v_2, \dots, v_n \in V$ astfel încât

$$V = \langle v_1, v_2, \dots, v_n \rangle = \{a_1 v_1 + a_2 v_2 + \dots + a_n v_n \mid a_1, a_2, \dots, a_n \in K\}.$$

Dacă V este finit generat, atunci nu există în V un sistem S infinit de vectori liniar independenți. Acest rezultat va fi o consecință a celor de mai jos.

Lema 1 (lema substituției). Fie V un spațiu vectorial finit generat, $v_1, v_2, \dots, v_n \in V$ astfel încât $V = \langle v_1, v_2, \dots, v_n \rangle$ și $v \in V$,

$v = a_1 v_1 + a_2 v_2 + \dots + a_n v_n$ cu $a_1, a_2, \dots, a_n \in K$. Presupunem că $a_j \neq 0$. Atunci

(1) $V = \langle v_1, \dots, v_{j-1}, v, v_{j+1}, \dots, v_n \rangle$, adică $v_1, \dots, v_{j-1}, v, v_{j+1}, \dots, v_n$ este, de asemenea, un sistem de generatori pentru V .

(2) Dacă $v_1, \dots, v_j, \dots, v_n$ formează o bază pentru V , atunci $v_1, \dots, v_{j-1}, v, v_{j+1}, \dots, v_n$ formează, de asemenea, o bază pentru V .

Teorema 1 (schimbului, Steintz). Fie K -spațiu vectorial V și vectorii $v_1, \dots, v_m, u_1, u_2, \dots, u_n \in V$ astfel încât $\text{ind}_K(v_1, \dots, v_m)$ și $V = \langle u_1, u_2, \dots, u_n \rangle$. Atunci $m \leq n$ și, mai puțin o eventuală renumerotare a vectorilor u_1, u_2, \dots, u_n , avem

$$V = \langle v_1, \dots, v_m, u_{m+1}, \dots, u_n \rangle.$$

Teorema 2. Orice spațiu vectorial V finit generat, diferit de spațiul nul, admite cel puțin o bază. Toate bazele sale sunt finite și au același număr de vectori.

Observație. Folosind lema lui Zorn se poate arăta că orice spațiu vectorial, nu neapărat finit generat, admite cel puțin o bază. De asemenea, se poate arăta că orice două baze ale unui spațiu vectorial au același cardinal, adică între vectorii acestora se poate stabili o corespondență funcțională bijectivă.

Dacă V este un K -spațiu vectorial nenul finit generat, numărul vectorilor dintr-o bază a lui V se numește **dimensiunea** lui V și se notează $\dim_K V$. Dacă $V = O = \{\emptyset\}$, atunci prin definiție $\dim_K V = 0$. Un spațiu vectorial care nu este finit generat nu poate avea bază finită; în acest caz spunem că V este spațiu vectorial de **dimensiune infinită** și scriem $\dim_K V = \infty$. Spațiile vectoriale de dimensiune finită sunt exact spațiile vectoriale finit generate.

Lema 2. Fie V un spațiu vectorial nenul de dimensiune finită și u_1, u_2, \dots, u_n un sistem finit de generatori pentru V . Atunci există o bază B a lui V astfel încât $B \subseteq \{u_1, u_2, \dots, u_n\}$. Altfel spus, din orice sistem (finit) de generatori se poate extrage o bază.

Observație. Folosind lema lui Zorn, rezultatul de mai sus poate fi probat și pentru spații vectoriale de dimensiune infinită.

Teorema 3 (a alternativei). Fie V un K -spațiu vectorial de dimensiune $n \in \mathbb{N}^*$ și $v_1, v_2, \dots, v_n \in V$. Sunt echivalente afirmațiile:

- (1) $B = \{v_1, v_2, \dots, v_n\}$ este o bază a lui V .
- (2) $V = \langle v_1, v_2, \dots, v_n \rangle$.
- (3) $\text{ind}_K(v_1, v_2, \dots, v_n)$.

Teorema 4. Fie V un K -spațiu vectorial de dimensiune $n \in \mathbb{N}^*$ și $v_1, v_2, \dots, v_m \in V$ astfel încât $\text{ind}_K(v_1, \dots, v_m)$, atunci există o bază B a lui V astfel încât $\{v_1, \dots, v_m\} \subseteq B$. Altfel spus, orice sistem de vectori liniar independenți poate fi completat până la o bază a lui V .

Corolar. Fie V un K -spațiu vectorial de dimensiune $n \in \mathbb{N}^*$ și $v \in V$, $v \neq 0$. Există o bază B a lui V astfel încât $v \in B$.

Teorema 5. Fie V un K -spațiu vectorial de dimensiune $n \in \mathbb{N}^*$.

- (1) Dacă X este un subspațiu vectorial al lui V , atunci X are dimensiunea finită și $\dim_K X \leq n$.
- (2) Dacă X și Y sunt subspații ale lui V , atunci $X \cap Y$ este subspațiu al lui V și

$$\dim_K(X + Y) = \dim_K X + \dim_K Y - \dim_K(X \cap Y).$$

- (3) Suma lui X cu Y este directă dacă și numai dacă

$$\dim_K(X + Y) = \dim_K X + \dim_K Y.$$

Corolar 1. Fie V un K -spațiu vectorial de dimensiune finită și X, Y două subspații ale lui V astfel încât $V = X \oplus Y$. Dacă B_1 este o bază a lui X , iar B_2 este o bază a lui Y , atunci $B_1 \cup B_2$ este o bază a lui V .

Corolar 2. Fie V un K -spațiu vectorial de dimensiune finită și X un subspațiu al lui V . Atunci există un subspațiu Y al lui V astfel încât $V = X \oplus Y$.

VI.4. Transformări liniare

Dacă V și V' sunt două spații vectoriale peste același corp comutativ K , atunci, ca și în cazul structurilor algebrice deja studiate, prezintă interes aplicațiile $f : V \rightarrow V'$ a căror acțiune este compatibilă cu operațiile de spațiu vectorial.

Definiție. Fie V și V' două spații vectoriale peste corpul K . O aplicație $f : V \rightarrow V'$ se numește **transformare liniară** dacă

- (1) $f(x+y) = f(x) + f(y)$, $\forall x, y \in V$,
- (2) $f(ax) = af(x)$, $\forall a \in K$, $\forall x \in V$.

Se observă că dacă $f : V \rightarrow V'$ este o transformare liniară, atunci

$$f\left(\sum_{j=1}^n a_j x_j\right) = \sum_{j=1}^n a_j f(x_j), \text{ oricare ar fi } a_1, \dots, a_n \in K \text{ și } x_1, \dots, x_n \in V.$$

Teorema 1. Fie V și V' două K -spații vectoriale și $f : V \rightarrow V'$ o transformare liniară. Avem:

- (1) $f(\theta) = \theta'$, $f(-x) = -f(x)$ oricare ar fi $x \in V$.
- (2) $\text{Ker}(f) \stackrel{\text{def}}{=} \{x \in V | f(x) = \theta'\}$ este subspațiu al lui V , iar $\text{Im}(f) \stackrel{\text{def}}{=} \{f(x) | x \in V\}$ este subspațiu al lui V' .
- (3) f este aplicație injectivă dacă și numai dacă $\text{Ker}(f) = O = \{\theta\}$.

Dacă $f : V \rightarrow V'$ este o transformare liniară, atunci $\text{Ker}(f)$ se numește **nucleul** lui f , iar $\text{Im}(f)$ se numește **imaginea** lui f .

Următorul rezultat arată că pe un spațiu vectorial V de dimensiune finită n se pot defini tot atâtea transformări liniare cu valori într-un spațiu vectorial dat V' , câte sisteme ordonate cu n vectori din V' există.

Teorema 2. Fie V un K -spațiu vectorial de dimensiune n și $B = (u_1, u_2, \dots, u_n)$ o bază a sa. Atunci, oricare ar fi K -spațiul vectorial V' și oricare ar fi vectorii $v'_1, v'_2, \dots, v'_n \in V'$ există o unică transformare liniară $f : V \rightarrow V'$ astfel încât $f(u_j) = v'_j$, $1 \leq j \leq n$.

Teorema 3. Fie V și V' două K -spații vectoriale și $f : V \rightarrow V'$ o transformare liniară de la V la V' . Dacă $\dim_K V = n < \infty$, atunci

$$\dim_K(\text{Ker}(f)) + \dim_K(\text{Im}(f)) = n = \dim_K V.$$

Definiție. O transformare liniară bijectivă se numește **izomorfism** de spații vectoriale. Dacă V și V' sunt două K -spații vectoriale, vom spune că V este **izomorf** cu V' , și scriem $V \sim V'$, dacă există un izomorfism $f : V \rightarrow V'$.

Vom spune că două K -spații vectoriale sunt de același **tip** dacă sunt izomorfe. Conform rezultatului următor, pentru orice $n \in \mathbb{N}$, există un singur tip de K -spațiu vectorial de dimensiune n ; dacă $n \geq 1$, atunci un prototip pentru K -spațiile vectoriale de dimensiune n este spațiul vectorial K^n .

Teorema 4. Dacă V este un K -spațiu vectorial de dimensiune $n \geq 1$, atunci $V \sim K^n$.

VI.5. Algebra operatorilor liniari ai unui spațiu vectorial de dimensiune finită

Există o legătură naturală între transformările liniare dintre două K – spații vectoriale de dimensiune finită și matricele cu coeficienți în corpul K . Vom defini mai întâi pentru transformări liniare operații care vor corespunde operațiilor cu matrice.

Fie transformările liniare $f : V \rightarrow V'$ și $g : V' \rightarrow V''$, unde V, V', V'' sunt spații vectoriale peste corpul comutativ K . Dacă $h = g \circ f$, atunci h este o transformare liniară de la V la V'' .

Dacă $f : V \rightarrow V'$ și $g : V \rightarrow V'$ sunt transformări liniare, atunci aplicația

$$f + g : V \rightarrow V', (f + g)(x) = f(x) + g(x)$$

este, de asemenea, o transformare liniară de la V la V' , numită **suma** lui f cu g .

În fine, dacă $a \in K$, iar $f : V \rightarrow V'$ este o transformare liniară între K – spațiile liniare V și V' , atunci aplicația

$$af : V \rightarrow V', (af)(x) = af(x)$$

este, de asemenea o transformare liniară.

Dacă V este un K – spațiu vectorial, o transformare liniară $f : V \rightarrow V$ se numește încă **operator liniar** pe V sau încă **endomorfism** al lui V . Vom nota cu $\text{End}_K(V)$ mulțimea tuturor operatorilor liniari pe V .

Dacă $f, g \in \text{End}_K(V)$ și $a \in K$, atunci evident $f + g$, $f \circ g$ și af aparțin lui $\text{End}_K(V)$. Așadar, dacă V este un spațiu vectorial peste corpul comutativ K , atunci pe mulțimea $\text{End}_K(V)$ avem definite operațiile

$$\text{End}_K(V) \times \text{End}_K(V) \rightarrow \text{End}_K(V), (f, g) \mapsto f + g,$$

$$\text{End}_K(V) \times \text{End}_K(V) \rightarrow \text{End}_K(V), (f, g) \mapsto f \circ g,$$

$$K \times \text{End}_K(V) \rightarrow \text{End}_K(V), (a, f) \mapsto af$$

și se poate demonstra că $\text{End}_K(V)$ are o structură de K – algebră în raport cu acestea.

Elementul zero al acestei K – algebrelor este operatorul **zero** $O : V \rightarrow V$, $O(x) = \theta$, elementul **unitate** este operatorul $1_V : V \rightarrow V$, $1_V(x) = x$.

După cum se va vedea în curând, în cazul când $\dim_K V = n < \infty$, verificarea axiomelor K – algebrei pentru $\text{End}_K(V)$ se poate face invocând faptul că $\mathbf{M}_n(K)$ are o structură de K – algebră în raport cu operațiile cu matrice.

Fie V un spațiu vectorial de dimensiune finită peste corpul comutativ K și $B = (u_1, u_2, \dots, u_n)$ o bază a lui V . Dacă f este un operator liniar pe V ,

imaginile $f(u_1), f(u_2), \dots, f(u_n)$ ale vectorilor bazei B prin f pot fi reprezentăți în mod unic ca combinații liniare cu coeficienți în K de u_1, u_2, \dots, u_n :

$$\begin{aligned}f(u_1) &= a_{11}u_1 + a_{21}u_2 + \dots + a_{n1}u_n = \sum_{i=1}^n a_{i1}u_i \\f(u_2) &= a_{12}u_1 + a_{22}u_2 + \dots + a_{n2}u_n = \sum_{i=1}^n a_{i2}u_i \\&\vdots \\f(u_n) &= a_{1n}u_1 + a_{2n}u_2 + \dots + a_{nn}u_n = \sum_{i=1}^n a_{in}u_i\end{aligned}$$

unde $a_{ij} \in K$, $1 \leq i, j \leq n$. Matricea $A = (a_{ij}) \in \mathbf{M}_n(K)$, unic determinată de operatorul f și de baza (ordonată!) B se numește **matricea asociată** în baza B operatorului f și o vom nota cu $M_B(f)$.

Teorema 1. Dacă V este un K -spațiu vectorial de dimensiune n și $B = (u_1, u_2, \dots, u_n)$ este o bază a sa, atunci aplicația

$$\varphi : \text{End}_K(V) \rightarrow \mathbf{M}_n(K), \quad \varphi(f) = M_B(f)$$

este bijectivă și

$$\begin{aligned}M_B(f+g) &= M_B(f) + M_B(g), \\M_B(f \circ g) &= M_B(f)M_B(g), \\M_B(af) &= aM_B(f),\end{aligned}$$

oricare ar fi $a \in K$ și $f, g \in \text{End}_K(V)$. Altfel spus, φ este izomorfism de K -algebri.

VI.6. Spațiul vectorial factor. Teorema fundamentală de izomorfism

Fie V un K -spațiu vectorial și N un subspațiu al lui V . Atunci N este un subgrup al grupului abelian $(V, +)$. În adevăr, dacă $x, y \in N$, atunci $x + y \in N$, iar dacă $x \in N$, atunci $-x = (-1)x \in N$. Cum orice subgrup al unui grup abelian este normal, putem considera grupul factor $\frac{V}{N}$ al grupului $(V, +)$ prin subgrupul N , elementele sale fiind clasele de resturi $\hat{v} = v + N$, $v \in V$ după subgrupul N , iar adunarea în grupul abelian $\left(\frac{V}{N}, +\right)$ fiind

$$\hat{u} + \hat{v} = \widehat{u+v}, \quad \forall \hat{u}, \hat{v} \in \frac{V}{N}.$$

Dacă $a \in K$ și $\hat{u} \in \frac{V}{N}$, atunci definim produsul $a\hat{u}$ prin

$$a\hat{u} \stackrel{\text{def}}{=} \widehat{au}$$

Definiția este corectă pentru că dacă $u' \in \hat{u}$, atunci $u' = u + x$ cu $x \in N$ și cum $ax \in N$, avem $au' - au = ax \in N$, deci $\widehat{au'} = \widehat{au}$. Evident operația

$$K \times \frac{V}{N} \rightarrow \frac{V}{N}, (a, \hat{u}) \mapsto a\hat{u}$$

verifică axiomele înmulțirii vectorilor cu scalari. Rezultă că $\frac{V}{N}$ are o structură de spațiu vectorial peste K și se numește **spațiu vectorial factor** al lui V prin subgrupul N .

Aplicația $\varphi : V \rightarrow \frac{V}{N}$, $\varphi(u) = \hat{u} = u + N$ este o transformare liniară

surjectivă de la V la $\frac{V}{N}$ pentru că

$$\varphi(u+v) = \widehat{u+v} = \hat{u} + \hat{v} = \varphi(u) + \varphi(v),$$

$$\varphi(au) = \widehat{au} = a\hat{u} = a\varphi(u)$$

oricare ar fi $u, v \in V$ și $a \in K$.

Teorema 1 (fundamentală de izomorfism). Fie V și V' două K -spații vectoriale și $f : V \rightarrow V'$ o transformare liniară. Atunci

$$\text{Im}(f) \simeq \frac{V}{\text{Ker}(f)}.$$

VII. DETERMINANȚI ȘI SISTEME DE ECUAȚII LINIARE

VII.1 Definiția determinanților. Proprietăți

Fie R un inel comutativ și $n \in \mathbb{N}^*$. Vom nota cu $R^n = \mathbf{M}_{n \times 1}(R)$. Dacă $A = (a_{ij}) \in \mathbf{M}_n(R)$, notăm cu c_j^A coloana j a matricei A ,

$$c_j^A = \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{nj} \end{pmatrix} \in R^n, 1 \leq j \leq n.$$

Dacă $x_1, x_2, \dots, x_n \in R^n$, notăm cu $[x_1, x_2, \dots, x_n]$ matricea X din $\mathbf{M}_n(R)$ a cărei coloană j este x_j , $1 \leq j \leq n$, adică $x_j = c_j^X$, $1 \leq j \leq n$.

În particular, dacă $A \in \mathbf{M}_n(R)$, atunci

$$A = [c_1^A, c_2^A, \dots, c_n^A].$$

Vom nota cu $e_1, e_2, \dots, e_n \in R^n$ coloanele matricei $I_n \in \mathbf{M}_n(R)$,

$e_j = c_j^{I_n}$, $1 \leq j \leq n$. Avem:

$$c_j^A = \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{nj} \end{pmatrix} = \begin{pmatrix} a_{1j} \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ a_{2j} \\ \vdots \\ 0 \end{pmatrix} + \dots + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ a_{nj} \end{pmatrix} = a_{1j}e_1 + a_{2j}e_2 + \dots + a_{nj}e_n = \sum_{i=1}^n a_{ij}e_i.$$

Spunem că o aplicație $f : \mathbf{M}_n(R) \rightarrow R$ este *n-liniară* dacă valorile sale depind *R-liniar* de coloanele matricelor $X = [x_1, \dots, x_n] \in \mathbf{M}_n(R)$, adică

$$f([x_1, \dots, x'_j + x''_j, \dots, x_n]) = f([x_1, \dots, x'_j, \dots, x_n]) + f([x_1, \dots, x''_j, \dots, x_n])$$

și

$$f([x_1, \dots, \lambda x_j, \dots, x_n]) = \lambda f([x_1, \dots, x_j, \dots, x_n])$$

oricare ar fi $x_1, \dots, x_j, x'_j, x''_j, \dots, x_n \in R^n$, $\lambda \in R$ și oricare ar fi j , $1 \leq j \leq n$.

Vom spune că o aplicație $f : \mathbf{M}_n(R) \rightarrow R$ este *alternată* dacă $f([x_1, x_2, \dots, x_n]) = 0$ ori de câte ori există $i \neq j$ astfel încât $x_i = x_j$. Altfel spus, f este alternată dacă $f(X) = 0$ oricare ar fi o matrice $X \in \mathbf{M}_n(R)$ cu două coloane egale.

Vom spune că o aplicație $f : \mathbf{M}_n(R) \rightarrow R$ este *strâmb-simetrică* dacă

$$f([x_1, \dots, x_i, \dots, x_j, \dots, x_n]) = -f([x_1, \dots, x_j, \dots, x_i, \dots, x_n])$$

oricare ar fi $x_1, x_2, \dots, x_n \in R^n$ și oricare ar fi i și j , $1 \leq i < j \leq n$. Altfel spus, dacă într-o matrice $X = [x_1, \dots, x_n] \in \mathbf{M}_n(R)$ permutează două coloane, atunci valoarea lui f își schimbă semnul.

Lema 1. *Orice aplicație $f : \mathbf{M}_n(R) \rightarrow R$ n-liniară și alternată este strâmb-simetrică.*

Teorema 1. Dacă $f : \mathbf{M}_n(R) \rightarrow R$ este o aplicație n -liniară și alternată, atunci oricare ar fi matricea $A = (a_{ij}) \in \mathbf{M}_n(R)$ avem

$$f(A) = \left(\sum_{\sigma \in S_n} \varepsilon_\sigma a_{\sigma(1)1} a_{\sigma(2)2} \dots a_{\sigma(n)n} \right) f(I_n).$$

Definiție. Fie R un inel comutativ și $A = (a_{ij}) \in \mathbf{M}_n(R)$. Prin definiție, **determinantul** matricei A , notat $\det(A)$, este

$$\det(A) = \sum_{\sigma \in S_n} \varepsilon_\sigma a_{\sigma(1)1} a_{\sigma(2)2} \dots a_{\sigma(n)n}.$$

Pentru determinantul lui A se folosește încă notația

$$|A| \text{ sau } \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}.$$

În continuare vom arăta că aplicația $D : \mathbf{M}_n(R) \rightarrow R$, $D(A) = |A|$, numită aplicația **determinant**, are proprietățile:

- (D₁) D este o aplicație n -liniară.
- (D₂) D este o aplicație alternată.
- (D₃) $D(I_n) = 1$.

Având în vedere și rezultatul de la teorema 1, va rezulta că D este unică aplicație n -liniară și alternată pe $\mathbf{M}_n(R)$ care ia valoarea 1 pe I_n . Ca o consecință a lui (D₁) și (D₂) avem și proprietatea:

- (D₄) D este aplicație strâmb-simetrică.

Teorema 2. Aplicația $D : \mathbf{M}_n(R) \rightarrow R$, $D(A) = |A|$, unde

$$|A| = \sum_{\sigma \in S_n} \varepsilon_\sigma a_{\sigma(1)1} a_{\sigma(2)2} \dots a_{\sigma(n)n},$$

oricare ar fi $A = (a_{ij}) \in \mathbf{M}_n(R)$ satisfacă condițiile (D₁), (D₂) și (D₃).

Teorema 3 (de dualitate). Oricare ar fi o matrice $A = (a_{ij}) \in \mathbf{M}_n(R)$, avem $|A^T| = |A|$, unde A^T este transpusa matricei A . În particular avem

$$|A| = \sum_{\sigma \in S_n} \varepsilon_\sigma a_{1\sigma(1)} a_{2\sigma(2)} \dots a_{n\sigma(n)}.$$

Observație. Proprietățile (D₁), (D₂) și (D₃) ale determinanților sunt formulate în termeni de coloane pentru matricele pătrate din $\mathbf{M}_n(R)$. Cum prin

transpunere liniile unei matrice pătrate A devin coloane ale matricei transpușe A^T și cum $|A| = |A^T|$ rezultă că:

(D'₁) Determinantul unei matrice $A \in \mathbf{M}_n(R)$ depinde R -liniar de fiecare linie a sa.

(D'₂) $|A| = 0$ dacă matricea A are două linii egale.

(D'₄) Dacă permuteam două linii ale lui A , determinantul matricei își schimbă semnul.

Teorema 4. Fie $A = (a_{ij}) \in \mathbf{M}_n(R)$. Atunci:

(D₅) Dacă la elementele unei coloane a lui A adunăm elementele altor coloane înmulțite cu un element $\lambda \in R$, valoarea determinantului matricei astfel obținute este egală cu $|A|$.

(D'₅) Dacă la elementele unei linii a lui A adunăm elementele altor linii înmulțite cu un element $\lambda \in R$, valoarea determinantului nu se schimbă.
Determinanții matricelor de ordin n se numesc încă **determinanți de ordin n** .

Observații.

1. Din analiza de mai sus rezultă că pentru o matrice $A = (a_{ij}) \in \mathbf{M}_n(R)$, R inel comutativ, determinantul matricei A este

$$|A| = \sum_{\sigma \in S_n} \varepsilon_\sigma a_{\sigma(1)1} a_{\sigma(2)2} \dots a_{\sigma(n)n}$$

sau, echivalent,

$$|A| = \sum_{\sigma \in S_n} \varepsilon_\sigma a_{1\sigma(1)} a_{2\sigma(2)} \dots a_{n\sigma(n)}.$$

Așadar $|A|$ este egal cu o sumă de $n!$ termeni. Fiecare termen este un produs de n coeficienți ai matricei A , câte unul, și numai câte unul din fiecare linie și fiecare coloană a lui A . Cum numărul permutărilor pare este egal cu numărul permutărilor impare, $\frac{n!}{2}$ termeni ai lui $|A|$ sunt precedați de semnul

"+" și $\frac{n!}{2}$ termeni sunt precedați de semnul "-".

2. Calculul determinantului folosind formulele de mai sus devine practic imposibil, chiar cu echipamente de calcul performante, atunci când ordinul determinantului este foarte mare. În secțiunile următoare vor fi elaborate metode eficiente de calcul pentru determinanți.

VII.2. Dezvoltarea unui determinant după elementele unei coloane (linii)

Fie R un inel comutativ, $n \in \mathbb{N}^*$ și $A = (a_{ij}) \in \mathbf{M}_n(R)$. Pentru orice i și j , $1 \leq i, j \leq n$, notăm cu A_{ij} matricea pătrată de ordin $n-1$ care se obține din A suprimând linia i și coloana j . Elementul $d_{ij} \in R$,

$$d_{ij} \stackrel{\text{def}}{=} (-1)^{i+j} |A_{ij}|$$

se numește **cofactoral** sau încă **complementul algebric** al lui a_{ij} . Matricea $A^* = (d_{ij})^T \in \mathbf{M}_n(R)$ se numește **adjuncta** (sau **matricea reciprocă**) a lui A .

Teorema 1. Dacă R este un inel comutativ și $A = (a_{ij}) \in \mathbf{M}_n(R)$, atunci

$$AA^* = A^*A = dI_n = \begin{pmatrix} d & 0 & \cdots & 0 \\ 0 & d & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d \end{pmatrix}$$

unde $d = |A|$.

Lema 1. Dacă $e_j = c_j^{I_n}$, $1 \leq j \leq n$, atunci oricare ar fi i și j , $1 \leq i, j \leq n$, avem:

$$\det([c_1^A, \dots, c_{j-1}^A, e_i, c_{j+1}^A, \dots, c_n^A]) = (-1)^{i+j} |A_{ij}|.$$

Lema 2. Fie R un inel comutativ și $A = (a_{ij}) \in \mathbf{M}_n(R)$. Atunci

$$(1) \quad |A| = \sum_{i=1}^n a_{ij} d_{ij}, \quad 1 \leq j \leq n$$

și

$$(2) \quad |A| = \sum_{j=1}^n a_{ij} d_{ij}, \quad 1 \leq i \leq n$$

adică, suma produselor dintre elementele coloanei j și cofactorii acestora este egală cu $|A|$ (dezvoltarea determinantului după elementele coloanei j), respectiv, suma produselor dintre elementele liniei i și cofactorii acestora este egală cu $|A|$ (dezvoltarea determinantului după elementele liniei i).

Lema 3. Dacă R este un inel comutativ și $A = (a_{ij}) \in \mathbf{M}_n(R)$, atunci

$$(1) \quad \sum_{i=1}^n a_{ij} d_{ik} = 0 \text{ oricare ar fi } j \neq k, \quad 1 \leq j, k \leq n$$

$$(2) \quad \sum_{j=1}^n a_{ij} d_{lj} = 0 \text{ oricare ar fi } i \neq l, 1 \leq i, l \leq n.$$

Altfel spus, suma produselor dintre elementele coloanei j și cofactorii elementelor coloanei k , $j \neq k$ este egală cu zero. De asemenea, este adevărat rezultatul corespunzător pentru linii.

Corolar. Fie R un inel comutativ, $A = (a_{ij}) \in \mathbf{M}_n(R)$ și $d = \det(A)$.

Dacă d este inversabil în inelul R , atunci A este inversabilă în inelul $\mathbf{M}_n(R)$ și $A^{-1} = d^{-1} A^*$.

Observație. În paragraful următor vom vedea că $|AB| = |A| \cdot |B|$ oricare ar fi $A, B \in \mathbf{M}_n(R)$. Dacă există $A^{-1} \in \mathbf{M}_n(R)$ astfel încât $AA^{-1} = A^{-1}A = I_n$, atunci $1 = |I_n| = |AA^{-1}| = |A| \cdot |A^{-1}|$. Cum $|A|, |A^{-1}| \in R$, rezultă că $|A|$ este inversabil în R . Așadar pentru o matrice $A \in \mathbf{M}_n(R)$ condiția necesară și suficientă să fie inversabilă în inelul $\mathbf{M}_n(R)$ este ca $|A|$ să fie inversabil în R . Matricele inversabile în $\mathbf{M}_n(\mathbb{Z})$ sunt cele de determinant ± 1 , matricele inversabile în $\mathbf{M}_n(\mathbb{R})$ sunt cele de determinant $\neq 0$, iar cele inversabile în $\mathbf{M}_n(\mathbb{Z}_8)$ sunt cele de determinant egal cu $\hat{1}, \hat{3}, \hat{5}$ sau $\hat{7}$. De asemenea, o matrice $A \in \mathbf{M}_n(K[X])$, K corp comutativ, este inversabilă în inelul $\mathbf{M}_n(K[X])$ dacă și numai dacă $|A| \in K^* = K \setminus \{0\}$.

VII.3. Alte rezultate asupra determinanților

Teorema 1. Fie R un inel comutativ și $A, B \in \mathbf{M}_n(R)$. Atunci $|AB| = |A| \cdot |B|$, adică determinantul produsului este egal cu produsul determinantelor.

Teorema 2. Oricare ar fi matricele $A = (a_{ij}) \in \mathbf{M}_m(R)$, $B \in \mathbf{M}_n(R)$ și $C \in \mathbf{M}_{m \times n}(R)$, avem

$$\begin{vmatrix} A & C \\ O & B \end{vmatrix} = |A| \cdot |B|.$$

Corolar 1. Dacă A_1, A_2, \dots, A_r cu $r \geq 2$ sunt matrice pătrate cu coeficienți în R , atunci

$$\begin{vmatrix} A_1 & & * \\ & A_2 & \ddots \\ O & & A_r \end{vmatrix} = |A_1| \cdot |A_2| \cdot \dots \cdot |A_r|,$$

unde în zona marcată cu "*" pot fi coeficienți arbitrazi din R .

Corolar 2. Dacă $A = \begin{pmatrix} a_1 & & * \\ & a_2 & \ddots \\ O & & a_n \end{pmatrix} \in \mathbf{M}_n(R)$ este o matrice superior triunghiulară, atunci

$$|A| = a_1 a_2 \dots a_n.$$

Observații.

- Folosind teorema de dualitate se arată imediat că

$$\begin{vmatrix} A & O \\ C & B \end{vmatrix} = |A| \cdot |B|,$$

oricare ar fi $A \in \mathbf{M}_m(R)$, $B \in \mathbf{M}_n(R)$, $C \in \mathbf{M}_{n \times m}(R)$ și

$$\begin{vmatrix} a_1 & & O \\ & a_2 & \ddots \\ * & & a_n \end{vmatrix} = a_1 a_2 \dots a_n.$$

- Folosind proprietățile (D'_5) și (D'_4) putem reduce orice matrice pătrată A din $\mathbf{M}_n(K)$, K corp comutativ, la o matrice T superior triunghiulară și $|A| = \pm |T|$ după cum s-a folosit un număr par (respectiv impar) de permutări de linii. Astfel, dacă $A \in \mathbf{M}_3(\mathbb{Z}_5)$,

$$A = \begin{pmatrix} \hat{0} & \hat{3} & \hat{1} \\ \hat{4} & \hat{2} & \hat{3} \\ \hat{3} & \hat{1} & \hat{2} \end{pmatrix},$$

atunci aplicând lui A operațiile: permutează prima linie cu a doua, apoi adunăm prima linie înmulțită cu $\hat{2}$ la a treia și în final adunăm linia a doua la a treia. Se obține

$$A = \begin{pmatrix} \hat{0} & \hat{3} & \hat{1} \\ \hat{4} & \hat{2} & \hat{3} \\ \hat{3} & \hat{1} & \hat{2} \end{pmatrix} \rightarrow \begin{pmatrix} \hat{4} & \hat{2} & \hat{3} \\ \hat{0} & \hat{3} & \hat{1} \\ \hat{3} & \hat{1} & \hat{2} \end{pmatrix} \rightarrow \begin{pmatrix} \hat{4} & \hat{2} & \hat{3} \\ \hat{0} & \hat{3} & \hat{1} \\ \hat{0} & \hat{2} & \hat{1} \end{pmatrix} \rightarrow \begin{pmatrix} \hat{4} & \hat{2} & \hat{3} \\ \hat{0} & \hat{3} & \hat{1} \\ \hat{0} & \hat{0} & \hat{2} \end{pmatrix} = T.$$

Cum s-a folosit o permutare de linii, avem

$$|A| = -|T| = -(\hat{4} \cdot \hat{3} \cdot \hat{2}) = -\hat{4} = \hat{1}.$$

Dacă $A = (a_{ij}) \in M_n(R)$, notăm cu l_i^A , linia i a matricei A ,

$$l_i^A = (a_{i1}, a_{i2}, \dots, a_{in}).$$

Teorema 3. Fie K un corp comutativ și $A = (a_{ij}) \in \mathbf{M}_n(K)$. Sunt echivalente afirmațiile:

- (1) $|A| \neq 0$.
- (2) $\text{ind}_K(c_1^A, c_2^A, \dots, c_n^A)$ în K -spațiul vectorial K^n al vectorilor coloană n -dimensionali.
- (3) $\text{ind}_K(l_1^A, l_2^A, \dots, l_n^A)$ în K -spațiul vectorial K^n al vectorilor linie n -dimensionali.

Fie acum K un corp comutativ. Ne preocupă problema să găsim n scalari $x_1, x_2, \dots, x_n \in K$ astfel încât să fie satisfăcute condițiile:

$$(S) \quad \begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases}$$

unde b_1, \dots, b_n și a_{ij} , $1 \leq i, j \leq n$ sunt elemente date din K . Ansamblul (S) de condiții se numește **sistem de n ecuații liniare în n necunoscute** x_1, x_2, \dots, x_n .

Matricea $A = (a_{ij}) \in \mathbf{M}_n(K)$ se numește **matricea sistemului** (S). Un sistem ordonat (x_1, \dots, x_n) de elemente din K care satisface fiecare din cele n ecuații ale lui (S) se numește **soluție** a sistemului. Dacă $|A| \neq 0$, atunci (S) se numește **sistem Cramer**.

Teorema 4. Fie K un corp comutativ și $A = (a_{ij}) \in \mathbf{M}_n(K)$. Sunt echivalente afirmațiile:

- (1) $|A| \neq 0$.
- (2) Oricare ar fi $b_1, b_2, \dots, b_n \in K$, sistemul

$$(S) \quad \begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases}$$

admite soluție unică.

Când $|A| \neq 0$, unica soluție a sistemului (S) este (x_1, x_2, \dots, x_n) , unde

$$x_j = |A|^{-1} \det([c_1^A, \dots, c_{j-1}^A, b, c_{j+1}^A, \dots, c_n^A]), \quad 1 \leq j \leq n$$

$$cu \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \in K^n \quad (\text{Regula lui Cramer}).$$

VII.4. Matrice și transformări elementare. Matrice eșalon

Fie R un inel comutativ. Notăm cu (e_{ij}) matricea din $\mathbf{M}_n(R)$ care în poziția (i, j) are coeficientul $1 \in R$ și în restul pozițiilor coeficientul $0 \in R$. Din regula de înmulțire a matricelor rezultă că

$$e_{ij}e_{st} = \begin{cases} e_{it} & \text{pentru } j = s \\ 0 & \text{pentru } j \neq s \end{cases} \quad (*)$$

Definim matricele

$$T_{ij}(a) = I_n + ae_{ij} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & a & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}, \quad i \neq j, \quad a \in R$$

$$P_{ij} = I_n - e_{ii} - e_{jj} + e_{ij} + e_{ji} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & 0 & & \\ & & & & 1 & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{pmatrix}, \quad 1 \leq i < j \leq n$$

$$M_i(u) = I_n - e_{ii} + ue_{ii} = i \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & u & & \\ & & & & 1 & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{pmatrix}, \quad 1 \leq i \leq n, u \in U(R)$$

unde $U(R)$ este mulțimea elementelor inversabile ale inelului R . Matricele $T_{ij}(a)$, P_{ij} și $M_i(u)$ definite mai sus se numesc **matrice elementare** respectiv de tip I, II și III.

Lema 1. Matricele elementare sunt inversabile în inelul $\mathbf{M}_n(R)$ și avem

$$T_{ij}(a)^{-1} = T_{ij}(-a), \quad P_{ij}^{-1} = P_{ij}, \quad M_i(u)^{-1} = M_i(u^{-1}).$$

Să observăm că dacă $A = (a_{ij}) \in \mathbf{M}_{m \times n}(R)$, iar $T_{ij}(a)$, P_{ij} și $M_i(u)$ sunt din $\mathbf{M}_m(R)$, atunci, invocând regula de înmulțire a matricelor, se constată:

- (I) $T_{ij}(a)A$ se obține din A adunând la linia i linia j înmulțită cu a ;
- (II) $P_{ij}A$ se obține din A permutează linia i cu linia j .
- (III) $M_i(u)A$ se obține din A înmulțind linia i cu u .

Multiplicările de mai sus vor fi numite **transformări elementare** respectiv de tip I, II și III asupra liniilor matricei A .

Definiție. O matrice $E \in \mathbf{M}_{m \times n}(R)$ se numește matricea eșalon (cu r pivoți, $1 \leq r \leq \min(m, n)$) dacă există $a_1, \dots, a_r \in R^* = R \setminus \{0\}$ astfel încât a_i , $1 \leq i \leq r$, este primul coeficient nenul din linia i a lui E , restul liniilor lui E conțin numai pe zero, iar coloanele j_1, j_2, \dots, j_r în care se găsesc respectiv a_1, a_2, \dots, a_r satisfac condiția $j_1 < j_2 < \dots < j_r$.

$$E = \begin{pmatrix} 0 \dots a_1 * & \dots & * \\ 0 & \dots & a_2 * & \dots & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & a_r * & \dots & * \\ 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 \end{pmatrix}$$

Teorema 1. Dacă K este un corp comutativ și $A = (a_{ij}) \in \mathbf{M}_{m \times n}(K)$, $A \neq O$, atunci există un număr finit de matrice elementare U_1, U_2, \dots, U_p de tip I sau II astfel încât,

$$U_p \dots U_2 U_1 A = E,$$

unde E este matrice eșalon.

Altfel spus A poate fi adusă la forma eșalon printr-un număr finit de transformări elementare de tip I sau II asupra liniilor sale.

Observație. Forma eșalon E a unei matrice pătrate $A \in \mathbf{M}_n(K)$, K corp comutativ, este o matrice superior triunghiulară și $|A| \neq 0$ dacă și numai dacă E are n pivoți a_1, \dots, a_n , aceștia trebuie să se găsească pe diagonala principală. Vom avea $|A| = (-1)^m |E| = (-1)^m a_1 a_2 \dots a_n$, unde m este numărul de permutări de linii folosite pentru găsirea formei eșalon. În caz contrar $|A| = 0$. În adevăr o transformare elementară de tip I conservă valoarea determinantului, iar o transformare de tip II schimbă semnul determinantului.

Teorema 2. Fie K un corp comutativ și $A = (a_{ij}) \in \mathbf{M}_n(K)$ astfel încât $|A| \neq 0$. Atunci există matricele elementare U_1, U_2, \dots, U_p de tip I, II sau III astfel încât

$$U_p U_{p-1} \dots U_1 A = I_n$$

$$\text{În acest caz } A^{-1} = U_p U_{p-1} \dots U_1.$$

Corolar. Orice matrice $A \in \mathbf{M}_n(K)$, K corp comutativ, cu $|A| \neq 0$ este produs finit de matrici elementare.

Aplicație: calculul inversei unei matrice.

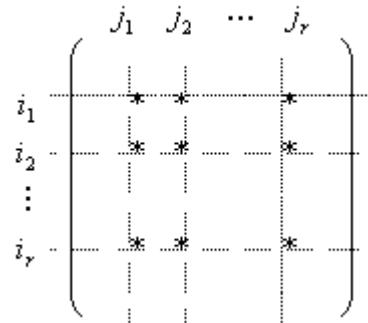
Presupunem că $A = (a_{ij}) \in \mathbf{M}_n(K)$, K corp comutativ și considerăm matricea $(A | I_n) \in \mathbf{M}_{n \times 2n}(K)$. Presupunem că $|A| \neq 0$, deci există A^{-1} . Cu notațiile de la teorema 2 avem $A^{-1} = U_p \dots U_2 U_1$. Dar

$$U_p \dots U_2 U_1 (A | I_n) = (U_p \dots U_2 U_1 A | U_p \dots U_2 U_1 I_n) = (I_n | A^{-1})$$

Așadar, dacă matricei $(A | I_n)$ îi aplicăm transformările elementare care reduc pe A la I_n , în al doilea compartiment apare A^{-1} .

VII.5. Rangul unei matrice

Peste tot în acest paragraf K este un corp comutativ. Dacă $A = (a_{ij}) \in \mathbf{M}_{m \times n}(K)$ și $r \in \mathbb{N}^*$, $r \leq \min(m, n)$, atunci cu coeficienții lui A care se găsesc la intersecțiile a r linii distințe $i_1 < i_2 < \dots < i_r$ și r coloane distințte $j_1 < j_2 < \dots < j_r$ putem forma o matrice pătrată M de ordin r numită **submatrice** a lui A .



Evident, putem forma $C_m^r \times C_n^r$ submatrice pătrate M de ordin r ale lui A ; determinanții unor asemenea submatrice ale lui A se numesc **minori** de ordin r ai lui A .

Definiție. Fie K un corp comutativ și $A = (a_{ij}) \in \mathbf{M}_{m \times n}(K)$, $A \neq O$.

Spunem că A are rangul r dacă A admite un minor nenul de ordin r și toți minorii lui A de ordin mai mare ca r sunt egali cu zero. Rangul matricei zero este prin definiție egal cu zero.

Dacă A are rangul r , folosin notația $\text{rang}(A) = r$.

Observații.

1. Dacă toți minorii de ordin k ai unei matrice $A \in \mathbf{M}_{m \times n}(K)$ sunt egali cu zero, atunci orice minor de ordin $k+1$ al lui A este egal cu zero. În adevăr, dezvoltând minorul de ordin $k+1$ după elementele unei linii (coloane), cofactorii care intervin într-o asemenea dezvoltare sunt egali cu zero căci, mai puțin eventual semnul, sunt minori de ordin k . Așadar o matrice A are rangul r dacă are cel puțin un minor nenul de ordin r și toți minorii de ordin $r+1$ sunt egali cu zero.
2. Dacă $E \in \mathbf{M}_{m \times n}(K)$ este o matrice eșalon cu r pivoți, atunci $\text{rang}(E) = r$. În adevăr, dacă pivoții a_1, a_2, \dots, a_r se găsesc în coloanele j_1, j_2, \dots, j_r , atunci submatricea M a lui E situată în primele r linii și coloanele j_1, j_2, \dots, j_r este de forma

$$M = \begin{pmatrix} a_1 & * & \cdots & * \\ & a_2 & \cdots & * \\ & & \ddots & \vdots \\ O & & & a_2 \end{pmatrix}$$

și $|M| = a_1 a_2 \dots a_r \neq 0$. Orice submatrice de ordin $r+1$ a lui E conține elemente din cel puțin una din ultimele $m-r$ linii ale lui E și deci determinantul ei este egal cu zero.

Teorema 1 (Kronecker). Fie K un corp comutativ, $A = (a_{ij}) \in \mathbf{M}_{m \times n}(K)$, $A \neq O$ și $r \in \mathbb{N}^*$. Sunt echivalente afirmațiile:

- (1) $\text{rang}(A) = r$;
- (2) Numărul maxim de coloane ale lui A liniar independente este r , adică $\dim_K \langle c_1^A, c_2^A, \dots, c_n^A \rangle = r$;
- (3) Numărul maxim de linii ale lui A liniar independente este r , adică $\dim_K \langle l_1^A, l_2^A, \dots, l_m^A \rangle = r$.

Observație. Dacă o matrice $A = (a_{ij}) \in \mathbf{M}_{m \times n}(K)$, admite un minor $|M| \neq 0$ de ordin r și toți cei $(m-r)(n-r)$ minori de ordin $r+1$ obținuți **bordând** pe M cu una din cele $m-r$ linii și una din cele $n-r$ coloane care nu au coeficienți comuni cu M sunt egali cu zero, atunci $\text{rang}(A) = r$. Sub ipotezele din teorema precedentă minorii de ordin $r+1$ care bordează pe M sunt

$$\left| \begin{array}{ccc|c} a_{11} & \cdots & a_{1r} & a_{1j} \\ \vdots & M & \vdots & \vdots \\ a_{r1} & \cdots & a_{rr} & a_{rj} \\ \hline a_{i1} & \cdots & a_{ir} & a_{ij} \end{array} \right|, \quad r < i \leq m, \quad r < j \leq n.$$

Pe baza observației de mai sus putem reduce volumul calculului necesar determinării rangului unei matrice.

S-a observat că rangul unei matrice eșalon se determină imediat, acesta fiind egal cu numărul pivoților. Importanța rezultatului din teorema 2 de mai jos este acum evident.

Lema 1. Fie V un K -spaiu vectorial și $v_1, v_2, \dots, v_m \in V$. Atunci oricare ar fi $a \in K^*$ și $i \neq j$, $1 \leq i, j \leq m$ avem

$$\begin{aligned} \langle v_1, \dots, v_i, \dots, v_j, \dots, v_m \rangle &= \langle v_1, \dots, av_i, \dots, v_j, \dots, v_m \rangle = \langle v_1, \dots, v_i + av_j, \dots, v_j, \dots, v_m \rangle = \\ &= \langle v_1, \dots, v_j, \dots, v_i, \dots, v_m \rangle \end{aligned}$$

Teorema 2. Rangul unei matrice $A \in \mathbf{M}_{m \times n}(K)$ nu se schimbă dacă asupra liniilor sale se efectuează transformări elementare. Altfel spus, dacă U este o matrice elementară de ordin m , atunci matricele UA și A au același rang.

VI.6. Sisteme de ecuații liniare. Metode de rezolvare

VI.6.1 Teorema Kronecker-Capelli. Metoda Gauss a eliminării

În paragraful VII.3. am considerat sisteme de n ecuații liniare, în n necunoscute cu coeficienți într-un corp comutativ K . Vom considera acum cazul general când numărul ecuațiilor nu coincide obligatoriu cu numărul necunoscitelor. Un sistem de m ecuații liniare în n necunoscute x_1, x_2, \dots, x_n cu coeficienți într-un corp comutativ K este un ansamblu (S) de condiții de forma

$$(S) \quad \begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m \end{cases}, \quad (1)$$

unde $a_{ij} \in K$, $1 \leq i \leq m$, $1 \leq j \leq n$ și $b_i \in K$, $1 \leq i \leq m$.

Scalarii $a_{ij} \in K$, $1 \leq i \leq m$, $1 \leq j \leq n$ se numesc **coeficienții** necunoscitelor, iar $b_i \in K$, $1 \leq i \leq m$ **termenii liberi** ai sistemului (S). Matricea $A = (a_{ij}) \in \mathbf{M}_{m \times n}(K)$ se numește matricea sistemului (S).

Dacă $b_1 = b_2 = \dots = b_m = 0$, spunem că sistemul (S) este **omogen**.

În continuare K^n și K^m reprezintă K – spațiul vectorial al vectorilor coloană n – dimensionali, respectiv m – dimensionali. Vom folosi notațiile

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in K^n, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \in K^m.$$

Matricea $\bar{A} = (A|b)$ din $M_{m \times (n+1)}(K)$ care se obține adăugând la A coloana b a termenilor liberi se numește **matricea extinsă** a sistemului (S).

Cu aceste convenții de notații, sistemul (S) poate fi scris sub forma matriceală

$$(S) \quad Ax = b \quad (2)$$

sau sub formă vectorială

$$(S) \quad c_1^A x_1 + c_2^A x_2 + \dots + c_n^A x_n = b \quad (3)$$

Reprezentarea de la (1) a sistemului (S) este **forma scalară**.

Sistemul ordonat (x_1, x_2, \dots, x_n) de scalari din K se numește **soluție** a sistemului (S) dacă verifică ansamblul de condiții de la (1), ceea ce revine la faptul că în K – spațiul vectorial K^m vectorul b se scrie ca o combinație liniară cu coeficienții x_1, \dots, x_n de coloanele $c_1^A, c_2^A, \dots, c_n^A \in K^m$ ale matricei A , sau încă vectorul $x \in K^n$ de componente x_1, x_2, \dots, x_n verifică condiția (2).

Spunem că sistemul (S) este **compatibil** dacă admite cel puțin o soluție; în caz contrar spunem că sistemul (S) este **incompatibil**. Spunem că sistemul (S) este **compatibil determinat** dacă admite soluție unică.

Am arătat în paragraful VII.3. că un sistem (S) de n ecuații liniare în n necunoscute este compatibil determinat dacă și numai dacă $|A| \neq 0$ și am precizat și cum poate fi determinată unică sa soluție (regula lui Cramer).

Pentru cazul general al sistemelor (S) de m ecuații liniare în n necunoscute, cu coeficienți într-un corp comutativ K , se impune:

(α) Să cunoaștem criterii cu ajutorul cărora să putem să testăm dacă un sistem este compatibil.

(β) Pentru sistemele compatibile să dăm o descriere adecvată a mulțimii tuturor soluțiilor.

(γ) Să prezintăm metode convenabile de rezolvare (de determinare a mulțimii tuturor soluțiilor) pentru sistemele compatibile.

Vom spune că două sisteme (S) și (S') de ecuații liniare sunt **echivalente** dacă admit aceleași soluții.

Lema 1. Dacă (S) $Ax = b$ este un sistem de m ecuații liniare în n necunoscute și $P \in \mathbf{M}_m(K)$, $|P| \neq 0$, atunci sistemul

$$(S') \quad PAx = Pb$$

este echivalent cu sistemul (S) .

Teorema 1 (Kronecker-Capelli). Sistemul

$$(S) \quad Ax = b$$

este compatibil dacă și numai dacă $\text{rang}(A) = \text{rang}(\bar{A})$.

Observație. Atunci când sistemul (S) $Ax = b$ este compatibil, acesta este echivalent cu sistemul (S'') . Pentru rezolvarea sistemului (S'') nu este necesar să folosim regula lui Cramer. Necunoscutelor x_j , $j \in \{1, 2, \dots, n\} \setminus \{j_1, j_2, \dots, j_r\}$ numite **necunoscute secundare** le dăm valori arbitrară din corpul comutativ K . În sistemul Cramer corespunzător în necunoscutele $x_{j_1}, x_{j_2}, \dots, x_{j_r}$, numite principale, se observă că necunoscuta x_{j_1} a fost eliminată din ecuația a două, x_{j_1} și x_{j_2} au fost eliminate din ecuația a treia și aşa mai departe. Această împrejurare permite să determinăm rapid valorile corespunzătoare pentru x_{j_1}, \dots, x_{j_r} făcând "marche arrière". Se determină x_{j_r} din ecuația r folosind faptul că $a_r \neq 0$. Se introduce valoarea găsită în ecuația $r-1$ și se determină $x_{j_{r-1}}$ folosind faptul că $a_{r-1} \neq 0$ și aşa mai departe.

Acet procedeu de rezolvare a sistemelor de ecuații liniare se numește **metoda Gauss a eliminării**.

VII.6.2. Soluțiile unui sistem compatibil de ecuații liniare

Fie (S) $Ax = b$ un sistem de m ecuații liniare în n necunoscute. Sistemul (S_0)

$$(S_0) \quad Ax = \mathbf{0} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in K^m$$

se numește **sistem omogen** asociat lui (S) .

Teorema 2. Mulțimea N_A a soluțiilor sistemului omogen (S_0) este un subspațiu liniar al lui K^n și

$$\dim_K(N_A) = n - \text{rang}(A).$$

Teorema 3. Fie (S) $Ax = b$ un sistem compatibil de ecuații liniare și $x^0 \in K^n$ o soluție particulară a sa. Dacă N_A este mulțimea soluțiilor sistemului omogen (S_0) asociat lui (S), atunci

$$x^0 + N_A \stackrel{\text{def}}{=} \left\{ x^0 + x \mid z \in N_A \right\}$$

este mulțimea tuturor soluțiilor sistemului (S).

Corolar. Un sistem (S) $Ax = b$ de m ecuații liniare în n necunoscute este compatibil determinat dacă și numai dacă $\text{rang}(A) = \text{rang}(\bar{A}) = n$.

Fie (S) $Ax = b$ un sistem compatibil de m ecuații liniare în n necunoscute, x^0 o soluție particulară a sa și $d = \dim_K(N_A)$. O bază $(x^{(1)}, x^{(2)}, \dots, x^{(d)})$ a spațiului N_A al soluțiilor sistemului omogen (S_0) asociat se numește **sistem fundamental** de soluții și avem

$$x^0 + N_A = \left\{ x^0 + \sum_{i=1}^d \lambda_i x^{(i)} \mid \lambda_1, \dots, \lambda_d \in K \right\}.$$

Așadar, pentru a rezolva un sistem (S) compatibil de ecuații liniare este suficient să găsim o soluție particulară x^0 a sa și un sistem fundamental de soluții pentru sistemul omogen (S_0) asociat.

VII.6.3. Metoda matriceală de rezolvare a sistemelor liniare de ecuații liniare

Fie sistemul compatibil (S) de m ecuații liniare în n necunoscute,

$$(S) \quad Ax = b.$$

Putem presupune că $m \leq n$ și că $\text{rang}(A) = m$. În adevăr fie $\bar{A} = (A|b)$ matricea extinsă a sistemului (S). Cum (S) este compatibil, avem $\text{rang}(\bar{A}) = \text{rang}(A) = r$. Așadar \bar{A} are r linii liniar independente și oricare altă linie a sa este combinație liniară de acestea. Presupunând, de exemplu, că primele r linii ale lui \bar{A} sunt liniar independente, atunci pentru orice i , $r < i \leq m$ există $\lambda_1, \dots, \lambda_r \in K$ astfel încât

$$l_i^{\bar{A}} = \lambda_1 l_1^{\bar{A}} + \lambda_2 l_2^{\bar{A}} + \dots + \lambda_r l_r^{\bar{A}}.$$

De aici rezultă că a i -a ecuație a sistemului (S) poate fi obținută înmulțind primele r ecuații respectiv cu $\lambda_1, \lambda_2, \dots, \lambda_r$ și adunându-le termen cu termen. Deducem că sistemul (S) este echivalent cu sistemul (S') format cu primele $r \leq n$ ecuații ale lui (S) și rangul matricei sistemului (S') este r , egal cu numărul ecuațiilor lui (S').

Așadar putem presupune $m \leq n$ și că $\text{rang}(A) = m$. Matricea A are deci o submatrixă pătrată B de ordin m astfel încât $|B| \neq 0$. Pentru a nu complica

notățiile, presupunem că B se găsește în primele m coloane ale lui A , deci matricea A se partajează astfel: $A = (B|S)$, unde $S \in M_{m \times d}(K)$, $d = n - m$. Submatricea B se numește bază a sistemului (S). Necunoscutele x_1, x_2, \dots, x_m corespunzătoare coloanelor lui B se numesc necunoscute principale (sau de bază), iar x_{m+1}, \dots, x_n se numesc necunoscute secundare. Folosim notățiile

$$x_B = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \in K^m, \quad x_S = \begin{pmatrix} x_{m+1} \\ x_{m+2} \\ \vdots \\ x_n \end{pmatrix} \in K^d, \quad x = \begin{pmatrix} x_B \\ x_S \end{pmatrix} \in K^n.$$

Având în vedere cum se înmulțesc matricele partionate în blocuri, sistemul (S) se mai scrie

$$(S) \quad Bx_B + Sx_S = b$$

și înmulțind la stânga cu B^{-1} , obținem

$$x_B = B^{-1}b - B^{-1}Sx_S.$$

Luând $x_S = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in K^d$, obținem $x_B = B^{-1}b \in K^m$ și acum este evident că

$$x^0 = \begin{pmatrix} B^{-1}b \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in K^n$$

este o soluție particulară a sistemului (S).

Sistemul omogen (S_0) asociat sistemului (S) se scrie

$$(S_0) \quad Bx_B + Sx_S = \mathbf{0}$$

și înmulțind la stânga cu B^{-1} , obținem

$$x_B = -B^{-1}Sx_S = Cx_S$$

unde $C \in \mathbf{M}_{m \times d}(K)$

$$C = -B^{-1}S = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1d} \\ c_{21} & c_{22} & \cdots & c_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{md} \end{pmatrix}$$

Luând $x_s = c_i^{I_d}$, $1 \leq i \leq d$, obținem soluțiile

$$x^{(1)} = \begin{pmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{m1} \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad x^{(2)} = \begin{pmatrix} c_{12} \\ c_{22} \\ \vdots \\ c_{m2} \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \dots, \quad x^{(d)} = \begin{pmatrix} c_{1d} \\ c_{2d} \\ \vdots \\ c_{md} \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

ale sistemului omogen (S_0) . Să observăm că $x^{(1)}, \dots, x^{(d)}$ sunt coloanele matricei

$$\begin{pmatrix} C \\ I_d \end{pmatrix} = \begin{pmatrix} c_{11} & \cdots & c_{1d} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{md} \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$

care are rangul d pentru că are minorul $|I_d| = 1 \neq 0$.

Așadar $\text{ind}_K(x^{(1)}, x^{(2)}, \dots, x^{(d)})$ și cum $\dim_K(N_A) = n - m = d$, rezultă că $x^{(1)}, x^{(2)}, \dots, x^{(d)}$ este o nază pentru N_A , adică un sistem fundamental de soluții. Mulțimea tuturor soluțiilor a sistemului (S) este deci

$$\begin{pmatrix} B^{-1}b \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \lambda_1 x^{(1)} + \dots + \lambda_d x^{(d)} \text{ cu } \lambda_1, \dots, \lambda_d \in K.$$

VII.6.4. *Minori caracteristici. Teorema lui Rouché*

Fie (S) $Ax = b$ un sistem de m ecuații liniare în n necunoscute. Presupunem că $\text{rang}(A) = r$ și fixăm un minor $|M|$ diferit de zero de ordin r al lui A , care are în continuare statutul de **minor principal**. Minorii de ordin $r+1$ ai matricei $\bar{A} = (A | b)$ obținuți bordând cu linii ale lui A din care nu provine M și cu coloana b a termenilor liberi se numesc **minori caracteristici** ai sistemului (S) .

Astfel, dacă M provine din primele r linii și r coloane ale lui A , minorii caracteristici sunt

$$\Delta_i \stackrel{\text{not}}{=} \begin{vmatrix} M & b_1 \\ \vdots & b_r \\ a_{i1} & \cdots & a_{ir} & b_i \end{vmatrix}, \quad i = r+1, \dots, n.$$

Evident, $\text{rang}(A) = \text{rang}(\bar{A})$ dacă și numai dacă $\Delta_i = 0$, $i = r+1, \dots, n$.

Așadar:

Teorema 4 (Rouché). *Un sistem (S) $Ax = b$ de ecuații liniare este compatibil dacă și numai dacă toți minorii caracteristici sunt egali cu zero.*

VIII. TEORIE JORDAN

VIII.1. Matrice canonică Jordan

Fie K un corp comutativ, V un spațiu vectorial de dimensiune n peste corpul K și $B = (e_1, \dots, e_n)$ o bază a lui V . Fie, de asemenea $e'_1, e'_2, \dots, e'_n \in V$ și $B' = (e'_1, \dots, e'_n)$. Există scalarii $p_{ij} \in K$ unic determinați astfel încât

$$e'_j = \sum_{i=1}^n p_{ij} e_i, \quad 1 \leq j \leq n.$$

Fie $P = (p_{ij}) \in \mathbf{M}_n(K)$.

Cu datele de mai sus avem:

Lema 1. $B' = (e'_1, \dots, e'_n)$ este o bază a lui V dacă și numai dacă $\det(P) \neq 0$.

Când $\det(P) \neq 0$, P se numește **matricea de trecere** de la baza B la baza B' .

Fie acum T un operator liniar definit pe spațiul vectorial V , adică o aplicație $T : V \rightarrow V$ astfel încât

$$\begin{aligned} T(x+y) &= T(x) + T(y) \\ T(\lambda x) &= \lambda T(x) \end{aligned}$$

oricare ar fi $x, y \in V$ și oricare ar fi $\lambda \in K$. Există $a_{ij} \in K$ unic determinați astfel încât

$$T(e_j) = \sum_{i=1}^n a_{ij} e_i, \quad 1 \leq j \leq n.$$

Dacă $A = (a_{ij}) \in \mathbf{M}_n(K)$, atunci A a fost numită matricea asociată operatorului T în baza B și am folosit notatia $A = M_B(T)$ (vezi paragraful 1.5.).

Teorema 1. Fie V un spațiu vectorial de dimensiune n peste corpul K , $B = (e_1, \dots, e_n)$ și $B' = (e'_1, \dots, e'_n)$ două baze ale lui V și $P = (p_{ij}) \in \mathbf{M}_n(K)$ matricea de trecere de la baza B la baza B' . Dacă $T : V \rightarrow V$ este un operator liniar, atunci

$$M_{B'}(T) = P^{-1} M_B(T) P.$$

Un operator liniar T definit pe spațiul vectorial V este complet determinat de modul cum acționează pe o bază a lui V . Pe de altă parte acțiunea operatorului T pe o bază poate fi descrisă cu ajutorul coeficienților matricei asociate lui T în baza dată și această descriere este cu atât mai simplă cu cât matricea asociată este mai apropiată de o matrice diagonală.

Înăind cont de modul cum se trece de la o bază a lui V la altă bază (vezi lema 1) și modul cum se modifică matricea asociată unui operator dat când se face o schimbare de bază, se impune să definim următoarea relație binară pe $\mathbf{M}_n(K)$:

Definiție. Fie K un corp comutativ și $A, B \in \mathbf{M}_n(K)$. spunem că matricea A este **asemenea** cu matricea B , și scriem $A \approx B$, dacă există $P \in \mathbf{M}_n(K)$ cu $\det(P) \neq 0$ astfel încât $P^{-1}AP = B$.

Dacă $n \in \mathbb{N}^*$ și $\lambda \in K$, matricea pătrată

$$J_n(\lambda) \stackrel{\text{def}}{=} \begin{pmatrix} \lambda & 1 & & & \\ & \lambda & 1 & & O \\ & & \lambda & 1 & \\ & & & \ddots & \ddots \\ O & & & & \ddots & 1 \\ & & & & & \lambda \end{pmatrix} \in \mathbf{M}_n(K)$$

se numește celula Jordan de ordin n asociată scalarului λ . Dacă $\lambda_1, \lambda_2, \dots, \lambda_s \in K$ și $n_1, n_2, \dots, n_s \in \mathbb{N}^*$ cu $n_1 + n_2 + \dots + n_s = n$, atunci

$$J_{n_1, n_2, \dots, n_s}(\lambda_1, \lambda_2, \dots, \lambda_s) \stackrel{\text{def}}{=} \begin{pmatrix} J_{n_1}(\lambda_1) & & & & \\ & O & & & \\ & & J_{n_2}(\lambda_2) & & \\ & & & \ddots & \\ & O & & & J_{n_s}(\lambda_s) \end{pmatrix} \in \mathbf{M}_n(K)$$

care are pe diagonală celulele Jordan $J_{n_1}(\lambda_1), J_{n_2}(\lambda_2), \dots, J_{n_s}(\lambda_s)$ iar pe restul pozițiilor pe $0 \in K$, se numește **matrice Jordan**.

Teorema 2. Dată o matrice pătrată $A \in \mathbf{M}_n(\mathbb{C})$, există o matrice Jordan J_A , unic determinată mai puțin ordinea celulelor Jordan de pe diagonală, astfel încât $A \approx J_A$.

Teorema 2'. Fie T un operator liniar definit pe un spațiu vectorial V de dimensiune n peste corpul \mathbb{C} . Există o bază B a lui V astfel încât $M_B(T)$ să fie matrice Jordan. Matricea Jordan astfel asociată operatorului T este unică și determinată mai puțin ordinea celulelor de pe diagonală.

VIII.2. Inele euclidiene

VIII.2.1. Relația de divizibilitate într-un domeniu de integritate

Fie R un domeniu de integritate, adică un inel comutativ cu $1 \neq 0$ și fără divizori ai lui zero. Vom avea în vedere în primul rând inelele de polinoame $K[X]$, K corp comutativ și inelul \mathbb{Z} al numerelor întregi. Relația de divizibilitate este cunoscută atât pentru numere întregi, cât și pentru polinoame. Este avantajos să definim și să studiem relația de divizibilitate în cadrul mai larg dat de domeniile de integritate.

Fie deci R un domeniu de integritate și a, b două elemente ale lui R . Spunem că b **divide** a și scriem $b|a$, dacă există $q \in R$ astfel încât $a = bq$; b se numește **divizor** al lui a , iar a se numește **multiplu** al lui b .

Relația de divizibilitate pe un domeniu de integritate R este o relație binară reflexivă și tranzitivă:

- (1) $a|a$, $\forall a \in R$ (**reflexivitate**),
- (2) dacă $a|b$ și $b|c$, atunci $a|c$ (**tranzitivitate**).

În adevăr, $a|a$, oricare ar fi $a \in R$ pentru că $a = a \cdot 1$. De asemenea, cum $a|b$ și $b|c$, există $q_1, q_2 \in R$ astfel încât $b = aq_1$ și $c = bq_2$, de unde $c = a(q_1q_2)$, deci $a|c$.

Să mai observăm că dacă $a|a_i$, $1 \leq i \leq n$ și $c_1, c_2, \dots, c_n \in R$, atunci $a \left| \sum_{i=1}^n a_i c_i \right.$, pentru că dacă $a_i = aq_i$, $1 \leq i \leq n$, atunci $\sum_{i=1}^n a_i c_i = aq$, unde $q = \sum_{i=1}^n q_i c_i$.

Dacă $a, b \in R$, spunem că a este **asociat în divizibilitate** cu b , și scriem $a \sim b$, dacă $a|b$ și $b|a$. Se verifică imediat că $a \sim b$ dacă și numai dacă

$b = au$ cu $u \in U(R)$, unde $(U(R), \cdot)$ este grupul multiplicativ al elementelor inversabile ale inelului R , numit grupul unităților lui R .

Definiția. Fie R un domeniu de integritate și a, b două elemente din R . Un element $d \in R$ se numește **cel mai mare divizor comun** al lui a și b , pe scurt **c.m.m.d.c.** al lui a și b , dacă:

- (a) $d | a$ și $d | b$,
- (b) dacă $c | a$ și $c | b$, atunci $c | d$.

Să observăm că dacă elementul $d' \in R$ satisfacă, de asemenea condițiile (a) și (b), atunci $d' | d$ și $d | d'$, deci $d' \sim d$, adică $d' = du$ cu $u \in U(R)$.

Așadar, c.m.m.d.c. al lui a și b , în caz că există, este unic determinat mai puțin o asociere în divizibilitate și se notează cu (a, b) (a nu se confunda cu perechea ordonată $(a, b)!$)

Teorema 1. Fie R un domeniu de integritate cu proprietatea că există c.m.m.d.c pentru oricare două elemente $a, b \in R$.

Dacă $a, b, c \in R$, atunci:

- (1) $((a, b), c) \sim (a, (b, c))$ (asociativitate),
- (2) $(ca, cb) \sim c(a, b)$,
- (3) dacă $(a, b) \sim 1$ și $(a, c) \sim 1$, atunci $(a, bc) \sim 1$,
- (4) dacă $a | bc$ și $(a, b) \sim 1$, atunci $a | c$,
- (5) dacă $a | c$, $b | c$ și $(a, b) \sim 1$, atunci $ab | c$.

În inelul \mathbb{Z} punând condiția $d \geq 0$, iar în inelul $K[X]$, K corp comutativ, punând condiția ca d să fie polinom unitar, rezultă că c.m.m.d.c. este unic determinat în sens strict. În aceste condiții în enunțul teoremei 1 putem înlocui simbolul " \sim " cu " $=$ ".

VIII.2.2. Definiția inelelor euclidiene

Definiție. Un domeniu de integritate R se numește **inel euclidian** dacă există o funcție $\varphi : R^* \rightarrow \mathbb{N}$, unde $R^* = R \setminus \{0\}$ astfel încât, oricare ar fi $a, b \in R$ cu $b \neq 0$, există $q, r \in R$ cu proprietățile

$$a = bq + r$$

și

$$\varphi(r) < \varphi(b) \text{ dacă } r \neq 0.$$

Elementele q și r se numesc **câțul**, respectiv **restul** împărțirii lui a prin b .

Inelul \mathbb{Z} este inel euclidian în raport cu funcția $\varphi: \mathbb{Z}^* \rightarrow \mathbb{N}$, $\varphi(a) = |a|$ căci după cum este știut pentru orice $a, b \in \mathbb{Z}$, $b \neq 0$ există $q, r \in \mathbb{Z}$ astfel încât

$$a = bq + r, |r| < |b|.$$

Următoarea teoremă stabilește faptul că inelul $K[X]$, K corp comutativ, este inel euclidian în raport cu funcția

$$\varphi: K[X]^* \rightarrow \mathbb{N}, \varphi(f) = \text{grad } f.$$

Teorema 2. Fie K un corp comutativ și $g \in K[X]$, $g \neq 0$. Oricare ar fi polinomul $f \in K[X]$ există polinoamele $q, r \in K[X]$ unic determinate astfel încât

$$f = gq + r$$

și

$$\text{grad } r < \text{grad } g \text{ dacă } r \neq 0.$$

Fie acum R un inel euclidian în raport cu funcția $\varphi: R^* \rightarrow \mathbb{N}$ și $a, b \in R^*$ astfel încât $\varphi(a) \geq \varphi(b)$.

Definim r_i , $i \geq 0$ prin $r_0 = a$, $r_1 = b$ și r_{i+1} este restul împărțirii lui r_{i-1} prin r_i dacă $i \geq 1$.

Pentru orice $i \geq 1$, fie q_i câtul împărțirii lui r_{i-1} prin r_i .

Atât timp cât $r_i \neq 0$, $i \geq 1$ putem efectua împărțirea cu rest a lui r_{i-1} prin r_i și cum

$$\varphi(r_0) \geq \varphi(r_1) > \varphi(r_2) > \varphi(r_3) > \dots$$

există n astfel încât $r_i \neq 0$ oricare ar fi $i \leq n$ și $r_{n+1} = 0$. Așadar avem următoarea secvență de împărțiri cu rest

$$\left\{ \begin{array}{ll} a = r_0 = r_1 q_1 + r_2 = b q_1 + r_2, & \varphi(r_2) < \varphi(r_1) = \varphi(b) \\ r_1 = r_2 q_2 + r_3, & \varphi(r_3) < \varphi(r_2) \\ \vdots & \vdots \\ r_{n-2} = r_{n-1} q_{n-1} + r_n, & \varphi(r_n) < \varphi(r_{n-1}) \\ r_{n-1} = r_n q_n + 0 & \end{array} \right. (*)$$

numită **algoritmul Euclid** pentru a și b .

Numărul n se numește **lungimea euclidiană** a perechii a, b de elemente din R .

Teorema 3. Fie R un inel euclidian în raport cu funcția $\varphi: R^* \rightarrow \mathbb{N}$ și $a, b \in R^*$ astfel încât $\varphi(a) \geq \varphi(b)$. Atunci c.m.m.d.c. al lui a și b este egal cu ultimul rest nenul din algoritmul Euclid pentru a și b . Mai mult, există $s, t \in R$, numiți **coeficienți Bézout** ai lui a și b , astfel încât $(a, b) = sa + tb$.

VIII.2.3. Aritmetică inelului $K[X]$

Fie K un corp comutativ, $a \in K$ și $f \in K[X]$. Conform teoremei restului (I.D. Ion, S. Bârza, L.Tufan, *Lecții de algebră, Fascicula I*, Ed. Fundației România de Mâine, București, 2004, capitolul 5, paragraful 5.6) există $q \in K[X]$ unic determinat astfel încât

$$f = (X - a)q + f(a),$$

de unde rezultă:

Teorema 4 (Rouché, a factorului). Fie K un corp comutativ, $a \in K$ și $f \in K[X]$. Atunci $X - a$ divide pe f dacă și numai dacă $f(a) = 0$, adică a este rădăcină a polinomului f .

Corolar. Fie K un corp comutativ, $f \in K[X]$ și $a, b \in K$, $a \neq b$. Atunci $(X - a)(X - b)$ divide pe f dacă și numai dacă $f(a) = 0$ și $f(b) = 0$.

Definiție. Fie K un corp comutativ și $f \in K[X]$ astfel încât $\text{grad } f = n > 0$. Spunem că f este **reductibil** peste K dacă există două polinoame $g, h \in K[X]$, de grade strict mai mici ca n astfel încât $f = g \cdot h$. În caz contrar spunem că f este **ireductibil** peste K .

Să observăm că două polinoame $f, g \in K[X]$ sunt asociate în divizibilitate dacă și numai dacă

$$g = a \cdot f \text{ cu } a \in K^* = K \setminus \{0\}$$

Dacă $f \in K[X]$ este ireductibil, atunci af este ireductibil oricare ar fi $a \in K^*$. De asemenea, dacă $f \in K[X]$ este ireductibil și $d \in K[X]$ este un divizor al lui f , atunci $d \sim 1$ sau $d \sim f$, adică $d = a$ sau $d = af$ cu $a \in K^*$.

Lemă. Fie K un corp comutativ și $f, g, h \in K[X]$ astfel încât $f \mid gh$. Dacă f este ireductibil peste K , atunci $f \mid g$ sau $f \mid h$.

Teorema 5. Fie K un corp comutativ și $f \in K[X]$ astfel încât $\text{grad } f = n > 0$. Atunci există polinoamele ireductibile p_1, p_2, \dots, p_m din $K[X]$ unic determinate mai puțin ordinea și o asociere în divizibilitate astfel încât

$$f = p_1 p_2 \dots p_m.$$

Observație. Reducând discuția precedentă doar la polinoamele monice din $K[X]$, rezultatul din teorema 5 devine: *orice polinom monic f de grad $n > 0$ din $K[X]$ se reprezintă în mod unic ca produs de polinoame monice ireductibile.*

Cum polinoamele monice ireductibile din $\mathbb{C}[X]$ sunt de forma $X - \lambda$ cu $\lambda \in \mathbb{C}$, rezultatul din teorema 5, restrâns la polinoamele monice din $\mathbb{C}[X]$, revine la: *orice polinom monic $f \in \mathbb{C}[X]$ de grad $n > 0$ poate fi scris în mod unic sub forma*

$$f = (X - \lambda_1)^{e_1} (X - \lambda_2)^{e_2} \dots (X - \lambda_t)^{e_t}$$

cum $\lambda_1, \dots, \lambda_t \in \mathbb{C}$ distincte și $e_1, e_2, \dots, e_t \in \mathbb{N}^*$ cu $e_1 + e_2 + \dots + e_t = n$.

De asemenea, cum polinoamele monice ireductibile din $\mathbb{R}[X]$ sunt de forma $X - a$ și $X^2 + bX + c$ cu $a, b, c \in \mathbb{R}$, $b^2 - 4c < 0$ rezultă că: *orice polinom monic $f \in \mathbb{R}[X]$ de grad $n > 0$ se reprezintă în mod unic sub forma*

$$f = (X - a_1)^{s_1} \dots (X - a_p)^{s_p} (X^2 + b_1 X + c_1)^{t_1} \dots (X^2 + b_q X + c_q)^{t_q}$$

cum $a_i, b_j, c_j \in \mathbb{R}$ pentru $1 \leq i \leq p$, $1 \leq j \leq q$, $b_j^2 - 4c_j < 0$ și $s_i, t_j \in \mathbb{N}^*$ astfel încât $s_1 + \dots + s_p + 2(t_1 + \dots + t_q) = n$.

VIII.3. Matrice aritmetic echivalente

Dacă R este un inel comutativ, vom nota cu $GL_n(R)$ grupul multiplicativ al matricelor inversabile din inelul $\mathbf{M}_n(R)$.

Am observat în paragraful VII.2. că o matrice $U \in \mathbf{M}_n(R)$ este inversabilă în $\mathbf{M}_n(R)$ dacă și numai dacă $\det(U)$ este element inversabil al inelului R .

Astfel avem

$$GL_n(\mathbb{Z}) = \left\{ U \in \mathbf{M}_n(\mathbb{Z}) \mid \det(U) = \pm 1 \right\}$$

și

$$GL_n(K[X]) = \left\{ U \in \mathbf{M}_n(K[X]) \mid \det(U) \in K^* \right\}$$

unde K este un corp comutativ.

Definiție. Fie (R, ϕ) un inel euclidian și $A, B \in \mathbf{M}_n(R)$. Spunem că matricea A este **aritmetic echivalentă** cu matricea B și scriem $A \sim B$, dacă există $U, V \in GL_n(R)$ astfel încât $UAV = B$.

Să observăm că relația binară " \sim " definită mai sus este o relație de echivalență pe $\mathbf{M}_n(R)$, adică

$$\begin{aligned} A \sim A, \forall A \in \mathbf{M}_n(R) & \text{ (reflexivitate),} \\ A \sim B \Rightarrow B \sim A & \text{ (simetrie),} \\ A \sim B \text{ și } B \sim C, \text{ atunci } A \sim C & \text{ (tranzitivitate).} \end{aligned}$$

În paragraful VII.4. au fost introduse matricele elementare $T_{ij}(a)$, P_{ij} și $M_i(u)$, respectiv de tip I, II și III și am observat că acestea aparțin grupului $GL_n(R)$. Dacă $A = (a_{ij}) \in \mathbf{M}_n(R)$ iar $T_{ij}(a)$, P_{ij} și $M_i(u)$ sunt matrice elementare de ordin n , atunci matricea

- (I) $AT_{ij}(a)$ se obține din A adunând la coloana j coloana i înmulțită cu $a \in R$
- (II) AP_{ij} se obține din A permutând coloana j cu coloana i
- (III) $AM_j(u)$ se obține din A înmulțind coloana j cu $u \in U(R)$.

ACESTE MULTIPLICĂRI VOR FI NUMITE TRANSFORMĂRI ELEMENTARE RESPECTIV DE TIP I, II și III ASUPRA COLOANELOR LUI A . ANALOG ÎN PARAGRAFUL VII.4. AM DEFINIT TRANSFORMĂRILE ELEMENTARE ASUPRA LINIILOR LUI A .

Definiție. Spunem că o matrice $D \in \mathbf{M}_n(R)$ are forma **diagonal-canonică** dacă

$$D = \begin{pmatrix} d_1 & & & O & \\ \ddots & & & & \\ & d_r & & 0 & \\ & & 0 & & \ddots \\ O & & & & 0 \end{pmatrix} \stackrel{\text{not}}{=} \text{diag}(d_1, d_2, \dots, d_r, \underbrace{0, \dots, 0}_{n-r \text{ ori}})$$

unde $d_1, d_2, \dots, d_r \in R^*$ și $d_1 | d_2 | \dots | d_r$.

Lema 1. Fie (R, φ) un inel euclidian și $A = (a_{ij}) \in \mathbf{M}_n(R)$, $A \neq 0$. Atunci există un număr finit de matrice elementare $U_1, U_2, \dots, U_p, V_1, V_2, \dots, V_q$ astfel încât

$$U_p \dots U_2 U_1 A V_1 V_2 \dots V_q = \text{diag}(d_1, d_2, \dots, d_r, 0, \dots, 0)$$

cu $d_1, d_2, \dots, d_r \in R^*$ și $d_1 | d_2 | \dots | d_r$.

Altfel formulat, matricea A poate fi adusă la forma diagonal-canonică efectuând un număr finit de transformări elementare asupra liniilor și coloanelor sale.

Teorema 1. Fie (R, ϕ) un inel euclidian și $A \in \mathbf{M}_n(R)$, $A \neq 0$. Atunci există $U, V \in GL_n(R)$ astfel încât

$$UAV = \text{diag}(d_1, d_2, \dots, d_r, 0, \dots, 0),$$

unde $d_1, d_2, \dots, d_r \in R^*$ și $d_1 | d_2 | \dots | d_r$. Numărul r și elementele d_1, d_2, \dots, d_r , mai puțin o asociere în divizibilitate, sunt unic determinate.

Când $R = \mathbb{Z}$ sau $R = K[X]$, K corp comutativ, cerând ca $d_i > 0$, $1 \leq i \leq r$, respectiv d_i polinom monic, $1 \leq i \leq r$, atunci d_1, d_2, \dots, d_r sunt unic determinate.

VIII.4. Matrice asemenea

Fie K un corp comutativ, $\mathbf{M}_n(K)$ inelul matricelor pătrate de ordin n cu coeficienți în K și $\mathbf{M}_n(K)[X]$ inelul polinoamelor în nedeterminata X cu coeficienți în $\mathbf{M}_n(K)$,

$$F = A_0 + A_1 X + A_2 X^2 + \dots + A_m X^m$$

cu $m \in \mathbb{N}$, $A_0, A_1, \dots, A_m \in \mathbf{M}_n(K)$.

Lema 1. Fie $F \in \mathbf{M}_n(K)[X]$, $F = A_0 + A_1 X + \dots + A_m X^m$ și $A \in \mathbf{M}_n(K)$. Există matricele $B_{m-1}, \dots, B_1, B_0, R \in \mathbf{M}_n(K)$ unic determinate astfel încât

$$F = (B_{m-1} X^{m-1} + \dots + B_1 X + B_0)(I_n X - A) + R. \quad (*)$$

$$\text{Mai mult, } R = F(A) = A_0 + A_1 A + \dots + A_m A^m.$$

Cum inelul $\mathbf{M}_n(K)$ nu este comutativ, avem încă o variantă a rezultatului de la lema 1, anume:

Lema 1*. În aceleași ipoteze ca la lema 1, există și sunt unic determinate matricele $C_{m-1}, \dots, C_1, C_0, R \in \mathbf{M}_n(K)$ astfel încât

$$F = (I_n X - A)(C_{m-1} X^{m-1} + \dots + C_1 X + C_0) + R.$$

$$\text{Mai mult, } R = A^m A_m + A^{m-1} A_{m-1} + \dots + A A_1 + A_0.$$

Remarcă. Orice polinom F din $\mathbf{M}_n(K)[X]$ poate fi scris ca o matrice din $\mathbf{M}_n(K[X])$ și reciproc.

Fie $A = (a_{ij}) \in \mathbf{M}_n(K)$. Atunci polinomul $I_n X - A \in \mathbf{M}_n(K)[X]$ poate fi scris și ca o matrice

$$I_n X - A = \begin{pmatrix} X - a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & X - a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & X - a_{nn} \end{pmatrix} \in \mathbf{M}_n(K[X])$$

și se numește **matricea caracteristică** a lui A .

Polinomul $p_A \in K[X]$,

$$p_A = \det \begin{pmatrix} X - a_{11} & \cdots & -a_{1n} \\ \vdots & \ddots & \vdots \\ -a_{n1} & \cdots & X - a_{nn} \end{pmatrix}$$

se numește **polinomul caracteristic** al matricei A .

Să observăm că printre cei $n!$ termeni ai determinantului $|I_n X - A|$ se găsește și

$$(X - a_{11})(X - a_{22}) \dots (X - a_{nn})$$

de unde rezultă că polinomul p_A are gradul n și că primii doi coeficienți sunt 1 și

$$-\sum_{i=1}^n a_{ii}. \text{ Cum ultimul coeficient este } p_A(0) = \det(-A) = (-1)^n |A| \text{ avem}$$

$$p_A = X^n - \left(\sum_{i=1}^n a_{ii} \right) X^{n-1} + \dots + (-1)^n |A| \in K[X].$$

Teorema 1 (Hamilton-Cayley). Oricare ar fi $A \in \mathbf{M}_n(K)$ avem $p_A(A) = \mathbf{0} \in \mathbf{M}_n(K)$. Cu alte cuvinte, orice matrice $A \in \mathbf{M}_n(K)$ este rădăcină a polinomului său caracteristic.

Teorema 2 (fundamentală a asemănării). Fie K un corp comutativ și $A, B \in \mathbf{M}_n(K)$. Sunt echivalente afirmațiile:

- (1) $A \approx B$,
- (2) $I_n X - A \sim I_n X - B$.

Așadar matricele A și B sunt asemenea dacă și numai dacă matricele lor caracteristice $I_n X - A$ și $I_n X - B$, gândite ca matrice cu coeficienți în inelul euclidian $K[X]$, sunt echivalente.

VIII.5. Forma canonica Jordan a unei matrice din $\mathbf{M}_n(\mathbb{C})$

Lema 1. Fie K un corp comutativ, $\lambda \in K$ și $n \in \mathbb{N}^*$. Avem

$$I_n X - J_n(\lambda) \sim \begin{pmatrix} 1 & & & \\ & \ddots & & O \\ & & 1 & \\ O & & & (X - \lambda)^n \end{pmatrix}.$$

Lema 2. Fie $\varphi, \psi \in K[X]$ două polinoame prime între ele, adică $c.m.m.d.c. (\varphi, \psi) = 1$. Atunci

$$\begin{pmatrix} \varphi & 0 \\ 0 & \psi \end{pmatrix} \sim \begin{pmatrix} 1 & 0 \\ 0 & \varphi\psi \end{pmatrix} \sim \begin{pmatrix} \varphi\psi & 0 \\ 0 & 1 \end{pmatrix}.$$

Lema 3. Fie celulele Jordan $J_{n_1}(\lambda_1), J_{n_2}(\lambda_2)$ și matricea Jordan J de ordin $n = n_1 + n_2$,

$$J = J_{n_1, n_2}(\lambda_1, \lambda_2) = \begin{pmatrix} J_{n_1}(\lambda_1) & 0 \\ 0 & J_{n_2}(\lambda_2) \end{pmatrix}.$$

Atunci

$$I_n X - J \sim \begin{pmatrix} 1 & & & & \\ & \ddots & & & 0 \\ & & 1 & & \\ & & & 0 & (X - \lambda_1)^{n_1} (X - \lambda_2)^{n_2} \end{pmatrix} \text{ dacă } \lambda_1 \neq \lambda_2$$

și

$$I_n X - J \sim \begin{pmatrix} 1 & & & & \\ & \ddots & & & 0 \\ & & 1 & & \\ & & & 0 & (X - \lambda_1)^{n_1} \\ & & & & (X - \lambda_2)^{n_2} \end{pmatrix} \text{ dacă } \lambda_1 = \lambda_2.$$

Rezultatul de la lema 3 poate fi generalizat. Fie $J = J_{n_1, \dots, n_s}(\lambda_1, \dots, \lambda_s)$ o matrice Jordan. Fie $\lambda_1, \dots, \lambda_t$ cu $t \leq s$ elementele distincte (eventual renumerotate) din lista $\lambda_1, \dots, \lambda_s$. Fie q_i numărul celulelor Jordan asociate lui λ_i , $1 \leq i \leq t$ de ordine respectiv

$$e_{i1} \geq e_{i2} \geq \dots \geq e_{iq_i}.$$

Avem $\sum_{i=1}^t q_i = s$ și $\sum_{i=1}^t \left(\sum_{j=1}^{q_i} e_{ij} \right) = n$ unde $n = n_1 + n_2 + \dots + n_s$ este

ordinul matricei Jordan J .

Polinoamele $(X - \lambda_i)^{e_{ij}}$, $1 \leq i \leq t$, $1 \leq j \leq q_i$ se aranjează în tabloul

$$\begin{cases} (X - \lambda_1)^{e_{11}}, & (X - \lambda_1)^{e_{12}}, & \dots, & (X - \lambda_1)^{e_{1q_1}} \\ (X - \lambda_2)^{e_{21}}, & (X - \lambda_2)^{e_{22}}, & \dots, & (X - \lambda_2)^{e_{2q_2}} \\ \vdots \\ (X - \lambda_t)^{e_{t1}}, & (X - \lambda_t)^{e_{t2}}, & \dots, & (X - \lambda_t)^{e_{tq_t}} \end{cases} \quad (*)$$

Fie $m = \max\{q_1, \dots, q_t\}$. Fie d_m, d_{m-1}, \dots, d_1 respectiv produsul polinoamelor din coloana unu, a doua, ..., a m -a a tabloului (*). Evident $d_1 | d_2 | \dots | d_m$ și $\text{grad } d_1 > 0$. Cu notațiile de mai sus avem:

Teorema 1.

$$I_n X - J_{n_1, \dots, n_s}(\lambda_1, \dots, \lambda_s) \sim \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & 0 \\ & & 1 & & & \\ & & & d_1 & & \\ & 0 & & & \ddots & \\ & & & & & d_m \end{pmatrix},$$

unde numărul valorilor 1 aflate pe diagonala principală este $n - m$.

Teorema 2. Fie $A \in \mathbf{M}_n(\mathbb{C})$. Există o matrice Jordan J_A unic determinată mai puțin ordinea celulelor de pe diagonală astfel încât $A \approx J_A$.

O matrice $A \in \mathbf{M}_n(\mathbb{C})$ este **diagonalizabilă** dacă $A \approx \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ cu $\lambda_i \in \mathbb{C}$ nu neapărat distințe. O matrice $N \in \mathbf{M}_n(\mathbb{C})$ este **nilpotentă** dacă există $m \in \mathbb{N}^*$ astfel încât $N^m = \mathbf{0}$.

Teorema 3. Oricare ar fi $A \in \mathbf{M}_n(\mathbb{C})$, există $D, N \in \mathbf{M}_n(\mathbb{C})$, D diagonalizabilă și N nilpotentă astfel încât $A = D + N$ și $DN = ND$.

VIII.6. Polinomul minimal al unei matrice

Fie K un subcorp al lui \mathbb{C} și $A \in \mathbf{M}_n(K)$. Există polinoame $f \in K[X]$, $f \neq 0$ astfel încât $f(A) = \mathbf{0} \in \mathbf{M}_n(K)$, de exemplu $f = p_A$ polinomul caracteristic al lui A (vezi teorema Hamilton-Cayley). Polinomul monic m_A de grad minim care admite pe A ca rădăcină se numește **polinomul minimal** al lui A .

Dacă $f(A) = \mathbf{0}$ cu $f \in K[X]$, atunci $m_A | f$ căci dacă $f = m_A q + r$ cu $r \neq 0$, $\text{grad } r < \text{grad } m_A$, atunci avem $f(A) = m_A(A)q(A) + r(A)$, deci $r(A) = 0$. Contradicție.

Observație. Fie K un subcorp al lui \mathbb{C} și $A \in \mathbf{M}_n(K)$ astfel încât ultimul factor invariant d_m al lui A se descompune în produs de puteri de factori liniari din $K[X]$, adică

$$d_m = (X - \lambda_1)^{e_1} \dots (X - \lambda_t)^{e_t}$$

cu $\lambda_1, \dots, \lambda_t \in K$ distincți și $e_1, \dots, e_t \in \mathbb{N}^*$. Atunci

- (1) Există o matrice Jordan $J \in \mathbf{M}_n(K)$ astfel încât $A \approx J$.
- (2) A este diagonalizabilă dacă și numai dacă $e_1 = \dots = e_t = 1$.

Teorema 1. Fie K un subcorp al lui \mathbb{C} și $A \in \mathbf{M}_n(K)$. Fie $d_1, d_2, \dots, d_m \in K[X]$ factorii invariante al lui A . Atunci:

- (1) $p_A = d_1 d_2 \dots d_m$, $m_A = d_m$.
- (2) (**teorema lui Frobenius**) m_A și p_A admit aceiași divizori ireductibili din $K[X]$.

VIII.7. Aplicații ale teoriei Jordan în studiul operatorilor liniari

Fie V un spațiu vectorial de dimensiune n peste \mathbb{C} , $T \in \text{End}_{\mathbb{C}}(V)$ un operator liniar definit pe V și $B = (e_1, \dots, e_n)$ o bază a lui V .

Dacă $A = M_B(T)$, definim **polinomul characteristic** p_T și **polinomul minimal** m_T al operatorului T prin $p_T = p_A$ și $m_T = m_A$.

Dacă $B' = (e'_1, \dots, e'_n)$ este o altă bază a lui V iar $P \in GL_n(\mathbb{C})$ este matricea de trecere de la B la B' și $B = M_{B'}(T)$, atunci $P^{-1}AP = B$, deci $A \approx B$. Rezultă că $I_n X - A \sim I_n X - B$ de unde obținem că $p_A = p_B$ și

$m_A = m_B$. Așadar definițiile polinoamelor p_T și m_T sunt corecte (nu depind de baza folosită).

Teorema 1. Fie V un spațiu vectorial de dimensiune finită n peste \mathbb{C} și $T \in \text{End}_{\mathbb{C}}(V)$. Există o bază B' a lui V astfel încât $M_{B'}(T)$, notată J_T , să fie matrice Jordan. Mai mult J_T este unic determinată mai puțin ordinea celulelor Jordan de pe diagonala sa.

Un subspațiu L al lui V este **invariant** în raport cu operatorul T dacă

$$\forall x \in L \Rightarrow T(x) \in L$$

Fie V un spațiu vectorial peste corpul comutativ K și $v \in V$, $v \neq 0$ iar $L = Kv$ subspațiul de dimensiune 1 generat de vectorul $v \neq 0$. Evident $T(L) \subseteq L$ dacă și numai dacă există $\lambda \in K$ astfel încât $T(v) = \lambda v$.

Definiție. Fie V un spațiu vectorial peste corpul comutativ K și T un operator liniar pe V . Un scalar $\lambda \in K$ se numește **valoare proprie** sau **valoare caracteristică** a operatorului T dacă există $v \in V$, $v \neq 0$ astfel încât $T(v) = \lambda v$. În acest caz v se numește **vector propriu** asociat valorii proprii λ .

Teorema 2. Fie K un corp comutativ, V un spațiu vectorial peste K de dimensiune finită n și $T \in \text{End}_K(V)$. Pentru un scalar $\lambda \in K$, următoarele afirmații sunt echivalente:

- (1) λ este valoare proprie a lui T .
- (2) $p_T(\lambda) = 0$.

Cu alte cuvinte, valorile proprii ale operatorului liniar T sunt exact rădăcinile din K ale lui p_T . Când $K = \mathbb{C}$, valorile proprii coincid cu rădăcinile lui p_T (obligatoriu din \mathbb{C}).

Remarcă. Determinarea vectorilor proprii $x = \sum_{i=1}^n x_i e_i$ corespunzători

valorii proprii λ revine la determinarea soluțiilor nebanale ale sistemului omogen $(*)$, unde $p_T(\lambda) = 0$, $\lambda \in K$.

IX. GRUPURI ABELIENE FINIT GENERATE

Așa cum se va vedea în continuare, în studiul grupurilor abeliene intervine în mod natural inelul euclidian \mathbb{Z} al numerelor întregi, aşa cum în studiul operatorilor liniari pe spații vectoriale a intervenit inelul euclidian $K[X]$, K corp comutativ.

IX.1. Grupuri abeliene libere de rang finit

Fie $(G,+)$ un grup abelian dat în notația aditivă. Pentru $k \in \mathbb{Z}$ și $x \in G$ definim

$$kx = \begin{cases} \underbrace{x+x+\dots+x}_{k \text{ ori}} & \text{pentru } k > 0 \\ 0 \in G & \text{pentru } k = 0 \in \mathbb{Z} \\ -((-k)x) & \text{pentru } k < 0 \end{cases}$$

Avem

- (1) $(h+k)x = hx + kx$,
 - (2) $k(x+y) = kx + ky$,
 - (3) $h(kx) = (hk)x$,
 - (4) $1 \cdot x = x$.
- oricare ar fi $x, y \in G$ și oricare ar fi $h, k \in \mathbb{Z}$.

Spunem că G este grup abelian **finit generat** dacă există un număr finit de elemente $x_1, x_2, \dots, x_n \in G$ astfel încât

$$G = \mathbb{Z}x_1 + \mathbb{Z}x_2 + \dots + \mathbb{Z}x_n = \{k_1x_1 + k_2x_2 + \dots + k_nx_n \mid k_i \in \mathbb{Z}\}.$$

În aceste condiții spunem că x_1, x_2, \dots, x_n este un **sistem (finit) de generatori** pentru grupul abelian $(G,+)$. Spunem că $B = (u_1, u_2, \dots, u_n) \subset G$ este o bază (sau încă o \mathbb{Z} -bază) pentru G dacă

- (a) $G = \mathbb{Z}u_1 + \mathbb{Z}u_2 + \dots + \mathbb{Z}u_n$,
 - (b) din $k_1u_1 + k_2u_2 + \dots + k_nu_n = 0$, unde $k_i \in \mathbb{Z}$ rezultă $k_1 = k_2 = \dots = k_n = 0$.
- adică u_1, u_2, \dots, u_n este un sistem de generatori pentru G liniar independenți peste \mathbb{Z} .

Fie $(G,+)$ un grup abelian și

$$2G = \{2x \mid x \in G\}.$$

Atunci $2G$ este un subgrup al lui G și cum G este abelian, putem considera grupul factor

$$\hat{G} = \frac{G}{2G} = \{\hat{x} = x + 2G \mid x \in G\}.$$

Grupul factor $\hat{G} = \frac{G}{2G}$ devine spațiu vectorial peste corpul \mathbb{Z}_2 dacă

punem

$$\bar{k}\hat{x} \stackrel{\text{def}}{=} \widehat{kx}, \quad \forall \bar{k} \in \mathbb{Z}_2, \quad \forall \hat{x} \in \hat{G}.$$

Lema 1. Fie $(F,+)$ un grup abelian care admite o \mathbb{Z} -bază (finită) $B = (u_1, u_2, \dots, u_n)$. Atunci $\hat{B} = (\hat{u}_1, \hat{u}_2, \dots, \hat{u}_n)$ este o \mathbb{Z}_2 -bază pentru \mathbb{Z}_2 -spațiul vectorial $\hat{F} = \frac{F}{2F}$.

Spunem că un grup abelian $(F,+)$ este **liber de rang n** dacă admite o \mathbb{Z} -bază $B = (u_1, u_2, \dots, u_n)$ cu n elemente.

Pentru orice $n \in \mathbb{N}^*$,

$$F = \mathbb{Z}^n = \{(x_1, x_2, \dots, x_n) \mid x_i \in \mathbb{Z}\}$$

este un grup abelian liber de rang n .

În adevăr, F este grup abelian în raport cu operația

$$(x_1, x_2, \dots, x_n) + (y_1, y_2, \dots, y_n) = (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n)$$

iar $B = (e_1, e_2, \dots, e_n)$, unde

$$e_1 = (1, 0, \dots, 0), e_2 = (0, 1, \dots, 0), \dots, e_n = (0, 0, \dots, 1)$$

este o \mathbb{Z} -bază pentru $F = \mathbb{Z}^n$.

Cum într-un spațiu vectorial toate bazele au același număr de vectori (egal cu dimensiunea spațiului), din lema 1 rezultă că într-un grup abelian liber F toate \mathbb{Z} -basele sale au același număr de elemente (egal cu rangul lui F).

În continuare dăm o caracterizare a \mathbb{Z} -bazelor unui grup abelian liber de rang n .

Lema 2. Fie $(F,+)$ un grup abelian liber de rang n , $B = (u_1, u_2, \dots, u_n)$ o \mathbb{Z} -bază a lui F , $u'_1, u'_2, \dots, u'_n \in F$ și $p_{ij} \in \mathbb{Z}$ astfel încât

$$u'_i = \sum_{j=1}^n p_{ij} u_j, \quad 1 \leq i \leq n.$$

Fie $P = (p_{ij}) \in \mathbf{M}_n(\mathbb{Z})$. Sunt echivalente afirmațiile.

- (1) $B' = (u'_1, u'_2, \dots, u'_n)$ este o \mathbb{Z} -bază pentru F .
- (2) P este matrice inversabilă în inelul $\mathbf{M}_n(\mathbb{Z})$.
- (3) $\det(P) = \pm 1$.

Lema 3. Fie $(F,+)$ un grup abelian liber de rang n , $B = (u_1, u_2, \dots, u_n)$ o \mathbb{Z} -bază a sa și $a \in \mathbb{Z}$. Atunci pentru orice $i \neq j$

$$B' = (u_1, \dots, u_i + au_j, \dots, u_n)$$

este o \mathbb{Z} -bază a lui F .

În teorema următoare se stabilește că subgrupurile unui grup abelian liber de rang n sunt grupuri abeliene libere de rang $m \leq n$. Mai precis:

Teorema 1. Fie $(F,+)$ un grup abelian liber de rang n și $H \neq 0$ un subgrup al lui F . Atunci H este grup abelian liber de rang m , $1 \leq m \leq n$ și există o \mathbb{Z} -bază (v_1, v_2, \dots, v_n) a lui F și $d_1, d_2, \dots, d_m \in \mathbb{N}^*$ cu $d_1 | d_2 | \dots | d_m$ astfel încât $(d_1 v_1, d_2 v_2, \dots, d_m v_m)$ este o \mathbb{Z} -bază a lui H .

Dacă $(G,+)$ este un grup abelian iar X, Y sunt subgrupuri ale lui G astfel încât

$$G = X + Y = \{x + y \mid x \in X, y \in Y\}$$

atunci următoarele afirmații sunt echivalente:

- (i) Dacă $x_1 + y_1 = x_2 + y_2$ cu $x_1, x_2 \in X$, $y_1, y_2 \in Y$, atunci $x_1 = x_2$ și $y_1 = y_2$.
- (ii) Dacă $x + y = 0$ cu $x \in X$, $y \in Y$, atunci $x = y = 0$.
- (iii) $X \cap Y = 0$.

În condițiile de mai sus spunem că G este sumă directă a lui X cu Y și scriem $G = X \oplus Y$.

Să observăm că dacă $G = X \oplus Y$, unde X și Y sunt subgrupuri abeliene libere de rang finit iar B_1 este o \mathbb{Z} -bază a lui X , B_2 o \mathbb{Z} -bază a lui Y , atunci $B = B_1 \cup B_2$ este o \mathbb{Z} -bază a lui G .

Considerațiile de mai sus pot fi extinse la un număr finit de subgrupuri ale lui G .

IX.2. Structura grupurilor abeliene finit generate

Fie $(G,+)$ un grup abelian. Reamintim că un element $x \in G$ are ordinul finit dacă există $k \in \mathbb{N}^*$ astfel încât $kx = 0$. Dacă x este element de ordin finit, atunci numărul $m = \min \{k \in \mathbb{N}^* \mid kx = 0\}$ se numește ordinul lui x și se folosește notația $\text{ord } x = m$. Dacă $\text{ord } x = m$ și $m = st$ cu $s, t \in \mathbb{N}^*$, atunci $\text{ord } sx = t$.

Dacă $x \in G$, atunci

$$\text{ann}_\mathbb{Z}(x) \stackrel{\text{def}}{=} \{k \in \mathbb{Z} \mid kx = 0\}$$

se numește **anulatorul** în \mathbb{Z} al lui x . Evident $\text{ann}_\mathbb{Z}(x)$ este un ideal al lui \mathbb{Z} , iar dacă $\text{ord } x = m \in \mathbb{N}^*$, atunci $\text{ann}_\mathbb{Z}(x) = m\mathbb{Z} = \{mq \mid q \in \mathbb{Z}\}$.

Dacă $(G_1,+), (G_2,+), \dots, (G_m,+)$ sunt grupuri abeliene și $G = G_1 \times G_2 \times \dots \times G_m$, atunci definind pe G **adunarea** pe componente

$$(x_1, x_2, \dots, x_m) + (y_1, y_2, \dots, y_m) \stackrel{\text{def}}{=} (x_1 + y_1, x_2 + y_2 + \dots + x_m + y_m)$$

se obține pe $G = G_1 \times G_2 \times \dots \times G_m$ o structură de grup abelian; grupul $(G,+)$ astfel introdus se numește **produsul direct** al grupurilor $(G_1,+)$, $(G_2,+)$, ..., $(G_m,+)$.

Teorema 1 (structura grupurilor abeliene finit generate). Fie $(G,+)$ un grup abelian finit generat. Atunci:

(1) există numerele $d_1, d_2, \dots, d_m, r \in \mathbb{N}^*$ cu $1 < d_1 \mid d_2 \mid \dots \mid d_m$ astfel încât

$$G \sim \mathbb{Z}_{d_1} \times \mathbb{Z}_{d_2} \times \dots \times \mathbb{Z}_{d_m} \times \mathbb{Z}^r,$$

adică grupul $(G,+)$ este izomorf ca produsul direct al grupurilor

$$(\mathbb{Z}_{d_1}, +), (\mathbb{Z}_{d_2}, +), \dots, (\mathbb{Z}_{d_m}, +), \underbrace{(\mathbb{Z}, +), \dots, (\mathbb{Z}, +)}_{r \text{ ori}}.$$

(2) Numerele r și d_1, d_2, \dots, d_m sunt unic determinate.

IX.3. Rezolvarea în numere întregi a sistemelor de ecuații liniare

Dacă (R, φ) este un inel euclidian și $A = (a_{ij}) \in \mathbf{M}_{m \times n}(R)$, $A \neq 0$, atunci există $U \in GL_m(R)$ și $V \in GL_n(R)$ astfel încât

$$UAV = \begin{pmatrix} d_1 & & & & \\ & \ddots & & & 0 \\ & & d_r & & \\ & & & 0 & \\ 0 & & & & \ddots \\ & & & & 0 \end{pmatrix}, \text{ cu } d_i \in R^* \text{ și } d_1 \mid d_2 \mid \dots \mid d_r.$$

În această formulare putem stabili o condiție necesară și suficientă ca un sistem de ecuații liniare cu coeficienți întregi să admită soluții întregi și să descriem convenabil mulțimea tuturor asemenea soluții.

Fie sistemul (S) de m ecuații liniare în n necunoscute x_1, \dots, x_n cu coeficienți întregi

$$(S) \quad \begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m \end{cases}$$

Folosind notațiile

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

sistemul (S) se scrie

$$(S) \quad Ax = b$$

Fie $U \in GL_m(\mathbb{Z})$ și $V \in GL_n(\mathbb{Z})$ astfel încât

$$UAV = \begin{pmatrix} d_1 & & & & 0 \\ & \ddots & & & \\ & & d_r & & \\ & & & 0 & \\ 0 & & & & \ddots \\ & & & & & 0 \end{pmatrix}, \quad 1 \leq d_1 \mid d_2 \mid \dots \mid d_r$$

și fie

$$b' = \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_m \end{pmatrix} = Ub$$

Sistemul (S) este echivalent cu sistemele (S')

$$(S') \quad UAx = Ub = b'$$

care mai poate fi scris

$$(S') \quad UAV \cdot V^{-1}x = b'$$

Acum se observă că sistemul (S') admite soluții întregi dacă și numai dacă $d_i \mid b'_i$, $1 \leq i \leq r$ și $b'_i = 0$ pentru $r < i \leq m$.

În aceste condiții o soluție particulară este

$$x^0 = V \begin{pmatrix} b'_1 / d_1 \\ b'_2 / d_2 \\ \vdots \\ b'_r / d_r \\ 0 \\ \vdots \\ 0 \end{pmatrix} = c_1^V \frac{b'_1}{d_1} + \dots + c_r^V \frac{b'_r}{d_r}$$

iar soluția generală este

$$x = x^0 + c_{r+1}^V \lambda_{r+1} + \dots + c_n^V \lambda_n \text{ cu } \lambda_{r+1}, \dots, \lambda_n \in \mathbb{Z}$$

(s-a notat cu c_j^V coloana j a matricei V).

Teorema 1. Fie $a_1, a_2, \dots, a_n \in \mathbb{Z}$ nu toate nule. Există

$V = (v_{ij}) \in GL_n(\mathbb{Z})$ astfel încât

$$(a_1, a_2, \dots, a_n)V = (d, 0, \dots, 0)$$

cu $d \in \mathbb{Z}$, $d > 0$. Numărul d este egal cu c.m.m.d.c. al numerelor a_1, a_2, \dots, a_n și

$$d = \sum_{i=1}^n a_i v_{i1}$$

Corolar. Ecuația

$$a_1 x_1 + a_2 x_2 + \dots + a_n x_n = b \text{ cu } a_1, a_2, \dots, a_n, b \in \mathbb{Z} \quad (*)$$

are soluții întregi dacă și numai dacă $d | b$, unde d este c.m.m.d.c. al lui a_1, a_2, \dots, a_n .

Mai mult, dacă

$$(a_1, a_2, \dots, a_n)V = (d, 0, \dots, 0) \text{ cu } V \in GL_n(\mathbb{Z})$$

și $d \in \mathbb{Z}$, $d > 0$ și $d | b$, atunci soluțiile întregi ale ecuației $(*)$ sunt

$$\frac{b}{d} c_1^V + \sum_{j=2}^n c_j^V \lambda_j \text{ cu } \lambda_2, \dots, \lambda_n \in \mathbb{Z}.$$

BIBLIOGRAFIE

1. Ion D. Ion, Bârză S., Tufan L., *Lecții de algebră, Fascicula I*, Editura Fundației România de Mâine, București 2004.
2. Ion D. Ion, Bârză S., Tufan L., *Lecții de algebră, Fascicula II (Algebră liniară)*, Editura Fundației România de Mâine, București 2005.
3. Albu T., Ion D. Ion, *Itinerar elementar în algebra superioară*, Editura ALL, București, 1997.
4. Artin M, *Algebra*, Prentice-Hall, New Jersey, 1991.
5. Childs L., *A Concrete Introduction To Higher Algebra*, Springer Verlag, 1979.
6. Cohn P.M., *Algebra*, Vol.II, John Wiley and Sons, 1977.
7. Ion D. Ion, Năstăsescu C., Niță C., *Complemente de algebră*, Editura Științifică și Enciclopedică, București, 1984.
8. Ion D. Ion, Niță C., Popescu D., Radu N., *Probleme de algebră*, Editura Didactică și Pedagogică, București, 1981.
9. C. Năstăsescu, C. Niță, C. Vraciu, *Bazele algebrei*, Editura Academiei, București, 1986.

ANALIZĂ MATEMATICĂ

Conf. univ dr. GHEORGHE GRIGORE

I. CORPUL NUMERELOR REALE

I.1. Mulțimi ordonate

Fie X și Y mulțimi nevide.

Definiție: *Mulțimea*

$$\{(x, y) \mid x \in X, y \in Y\}$$

se numește *produs scalar al mulțimii X cu mulțimea Y* și se notează $X \times Y$.

Definiție. Se numește *relație de ordine pe o mulțime nevidă X orice submulțime $D \subset X \times X$ având proprietățile:*

- i) $(x, x) \in D$ pentru orice $x \in X$,
- ii) dacă $(x, y) \in D$ și $(y, x) \in D$, atunci $x = y$,
- iii) dacă $(x, y) \in D$ și $(y, z) \in D$, atunci $(x, z) \in D$.

Se scrie $x \leq y$ dacă $(x, y) \in D$ iar proprietățile precedente se numesc reflexivitatea, antisimetria respectiv tanzitivitatea relației de ordine. Dacă pe X s-a definit o relație de ordine se spune că X este o mulțime ordonată.

Fie X o mulțime ordonată și $A \subset X$, $A \neq \emptyset$.

Definiție. *Mulțimea A se numește majorată dacă există $x \in X$ astfel încât $a \leq x$ pentru orice $a \in A$. Elementul x se numește **majorant** pentru mulțimea A .*

*Mulțimea A se numește minorată dacă există $x \in A$ astfel încât $x \leq a$ pentru orice $a \in A$. Elementul x se numește **minorant** pentru mulțimea A .*

*O mulțime care este minorată și majorată se numește **mulțime mărginită**.*

Dacă $x \in A$ și este majorant (minorant) pentru mulțimea A se scrie $x = \max A$ ($x = \min A$) și se spune că este cel mai mare (mic) element al mulțimii A .

Definiție. *Cel mai mic majorant al mulțimii majorate A se numește **margine superioară** a mulțimii A și se notează $\sup A$. Dacă A este minorată, cel mai mare minorant al ei se numește **marginea inferioară** pentru A și se notează $\inf A$.*

Observație. Nu pentru orice mulțime majorată (minorată) există margine superioară (inferioară).

Definiție. *Mulțimea ordonată X se numește **completă ordonată** dacă pentru orice submulțime nevidă și majorată există margine superioară. Se spune că **ordinea este completă**.*

Într-o mulțime complet ordonată orice submulțime nevidă și minorată admite margine inferioară.

Definiție. Mulțimea ordonată X se numește **total ordonată** dacă pentru orice pereche $(x, y) \in X \times X$ avem $x \leq y$ sau $y \leq x$. Se spune că **ordinea este totală**.

I.2. Corpuri complet ordonate

Definiție. Se numește **corp ordonat** orice corp comutativ K înzestrat cu o relație de ordine totală astfel încât:

- i) Pentru orice $x, y, z \in K$, $x \leq y$ avem $x + z \leq y + z$,
- ii) Dacă $x, y \in K$, $0 \leq x$, $0 \leq y$, atunci $0 \leq xy$.

Vom nota $x < y$ dacă $x \leq y$ și $x \neq y$.

Definiție. Se numește **corp complet ordonat** orice corp ordonat în care ordinea este completă.

Definiție. Se numește **corp al numerelor reale** orice corp complet ordonat.

Fie R un corp al numerelor reale.

Definiție. Mulțimea $A \subset R$ se numește **inductivă** dacă pentru orice $x \in A$ avem $x + 1 \in A$.

Notăm $\mathcal{A} = \{A \subset R \mid A \text{ inductivă}, 0 \in A\}$.

Definiție. Mulțimea $N = \bigcap_{A \in \mathcal{A}} A$ se numește **mulțimea numerelor naturale**

din R .

Teorema (Principiul inducției complete). Fie R un corp al numerelor reale și $N \subset R$ mulțimea numerelor naturale. Dacă $A \subset N$ este o mulțime inductivă care conține pe 0, atunci $A = N$.

Corolar 1. Suma și produsul a două numere naturale este un număr natural.

Corolar 2.

- i) Dacă $n \in N$, $n \neq 0$, atunci $n - 1 \in N$
- ii) Dacă $n \in N$ atunci $\{x \in N \mid n < x < n + 1\} = \emptyset$
- iii) $n + 1 = \min \{x \in N \mid n < x\}$
- iv) dacă $A \subset N$, $A \neq \emptyset$, atunci există $\min A$.
- v) Dacă $m, n \in N$, $n < m$, atunci există $r \in N$ astfel încât $m = n + r$.

Definiție. Mulțimea $Z = N \cup \{-n \mid n \in N\}$ se numește **mulțimea numerelor întregi** din R .

Propoziție. $(Z, +, \cdot)$ este inel comutativ cu unitate.

Definiție. Mulțimea $Q = \left\{ mn^{-1} \mid m, n \in \mathbb{Z}, n \neq 0 \right\}$ se numește **mulțimea numerelor raționale** din \mathbb{R} .

Propoziție. $(Q, +, \cdot)$ este corp ordonat.

Definiție. Numerele reale care nu sunt raționale se numesc **iraționale**.

Teoremă. Pentru orice $x \in \mathbb{R}$, $x > 0$ și pentru orice $n \in \mathbb{N}$, $n \geq 1$ există și este unic $y \in \mathbb{R}$, $y > 0$ astfel încât $x = y^n$. Se notează $y = \sqrt[n]{x}$.

Propoziție (Principiul lui Arhimede). Pentru orice $x \in \mathbb{R}$ și orice $y \in \mathbb{R}$, $y > 0$ există $n \in \mathbb{N}$ astfel încât $x < ny$.

Corolar.

- i) Dacă $a \in \mathbb{R}$, $a > 0$, atunci există $n \in \mathbb{N}$ astfel încât $\frac{1}{n} < a$
- ii) dacă $a \in \mathbb{R}$, $a \geq 0$ și $a \leq \frac{1}{n}$ pentru orice $n \in \mathbb{N} \setminus \{0\}$, atunci $a = 0$.
- iii) dacă $a, b \in \mathbb{R}$, $a < b$, atunci există $r \in Q$ astfel încât $a < r < b$.
- iv) dacă $a \in \mathbb{R}$ atunci $a = \sup \{t \in Q \mid t < a\}$ și $a = \inf \{t \in Q \mid a < t\}$.
- v) dacă $a \in \mathbb{R}$ atunci există și este unic $n \in \mathbb{Z}$ astfel încât $n \leq a < n+1$. Se notează $n = [a]$ și se numește partea întreagă a lui a .

Definiție. Corpurile ordonate R' și R'' se numesc izomorfe dacă există $f : R' \rightarrow R''$ o bijecție aditivă, multiplicativă și care păstrează ordinea.

Teoremă. Orice corpuri de numere reale sunt izomorfe.

Se vorbește atunci despre "corpul numerelor reale" și se notează \mathbb{R} (respectiv \mathbb{N} , \mathbb{Z} , \mathbb{Q}).

I.3. Proprietăți topologice în corpul numerelor reale

Definiție. Mulțimea $I \subset \mathbb{R}$ se numește interval dacă pentru orice $x, y \in I$ și orice $z \in \mathbb{R}$ cu $x \leq z \leq y$ avem $z \in I$.

Observație. O mulțime $I \subset \mathbb{R}$ este interval dacă și numai dacă este de forma (a, b) , $[a, b]$, $[a, b)$, $(a, b]$, $[a, \infty)$, (a, ∞) , $(-\infty, a]$, $(-\infty, a)$, \mathbb{R} .

Teoremă (Principiul lui Cantor). Intersecția oricărui sir descrescător de intervale nevide și închise este nevidă.

Definiție. Mulțimea V se numește vecinătate a numărului a dacă există un interval deschis astfel încât $a \in I \subset V$.

Definiție. Numărul a se numește punct de acumulare pentru mulțimea $A \subset \mathbb{R}$ dacă pentru orice vecinătate V a lui a avem: $(V \setminus \{a\}) \cap A \neq \emptyset$.

Propoziție. Numărul a este punct de acumulare pentru mulțimea A dacă pentru orice vecinătate V a lui a mulțimea $V \cap A$ este infinită.

Exemple.

1. Orice număr real x este punct de acumulare pentru mulțimea \mathbb{Q} a numerelor raționale.
2. O mulțime finită nu admite puncte de acumulare.
3. \mathbb{Z} nu admite puncte de acumulare.
4. Orice punct din $[a,b]$ ($a < b$) este punct de acumulare pentru (a,b) . Dacă $x \notin [a,b]$, atunci x nu este punct de acumulare pentru $[a,b]$.
5. 0 (zero) este punct de acumulare pentru $\left\{ \frac{1}{n} \mid n \in \mathbb{N}^* \right\}$.

Teoremă (Bolzano-Weierstrass). Pentru orice mulțime infinită și mărginită există cel puțin un punct de acumulare.

Pentru $A \subset \mathbb{R}$ vom nota cu A' mulțimea punctelor de acumulare ale lui A .

Definiție. Mulțimea $A \subset \mathbb{R}$ se numește închisă dacă $A' \subset A$.

Teoremă (Borel-Lebesgue). O submulțime din \mathbb{R} este închisă și mărginită dacă și numai dacă din orice acoperire a sa cu intervale deschise se poate extrage o subacoperire finită.

II. ȘIRURI DE NUMERE REALE

II.1. Șiruri convergente. Definiție, proprietăți

Definiție. Se numește șir de numere reale orice funcție $f : \mathbb{N} \rightarrow \mathbb{R}$. Elementul $x_n = f(n)$ se numește termenul general (de rang n), iar șirul se notează $(x_n)_{n \in \mathbb{N}}$.

Dacă pentru $k \in \mathbb{N}$, $k \geq 1$, valorile funcției f se notează $f(j) = x_{k+j}$, $j \in \mathbb{N}$, atunci pentru șirul f se folosește notația $(x_n)_{n \geq k}$. De exemplu, $\left(\frac{1}{n} \right)_{n \geq 1}$.

Definiție. Șirul $(x_n)_{n \in \mathbb{N}}$ se numește convergent dacă există $x \in \mathbb{R}$ și pentru orice $\varepsilon > 0$ există $n_0 \in \mathbb{N}$ astfel încât pentru orice $n \in \mathbb{N}$, $n \geq n_0$ avem $|x_n - x| < \varepsilon$.

Numărul x se numește limita șirului și se notează $\lim_{n \rightarrow \infty} x_n = x$.

Propoziția 1. Limita unui șir convergent este unică.

Propoziția 2. *Şirul $(x_n)_{n \in \mathbb{N}}$ este convergent dacă și numai dacă există $x \in \mathbb{R}$ și pentru orice vecinătate V a lui x există $n_0 \in \mathbb{N}$ astfel încât pentru orice $n \geq n_0$, $x_n \in V$. (Se spune pe scurt că în afara oricărei vecinătăți a lui x se află cel mult un număr finit de termeni ai şirului).*

Definiție. *Şirul $(x_n)_{n \in \mathbb{N}}$ se numește mărginit dacă există numerele reale a, b astfel încât $x_n \in [a, b]$ pentru orice $n \in \mathbb{N}$.*

Observație. *Şirul $(x_n)_{n \in \mathbb{N}}$ este mărginit dacă și numai dacă $\exists M > 0$ astfel încât $|x_n| \leq M$ pentru orice $n \in \mathbb{N}$.*

Şirul $(x_n)_{n \in \mathbb{N}}$ este mărginit dacă și numai dacă mulțimea $\{x_n | n \in \mathbb{N}\}$ este mărginită.

Propoziția 3. *Orice şir convergent este mărginit.*

Observație. *Orice şir nemărginit nu este convergent.*

Teorema 1 (operații algebrice cu şiruri convergente). *Fie $(x_n)_{n \in \mathbb{N}}$ și $(y_n)_{n \in \mathbb{N}}$ două şiruri convergente. Atunci*

a) *Şirul $(x_n + y_n)_{n \in \mathbb{N}}$ este convergent și $\lim_{n \rightarrow \infty} (x_n + y_n) = \lim_{n \rightarrow \infty} x_n + \lim_{n \rightarrow \infty} y_n$,*

b) *Şirul $(x_n y_n)_{n \in \mathbb{N}}$ este convergent și $\lim_{n \rightarrow \infty} (x_n y_n) = \lim_{n \rightarrow \infty} x_n \lim_{n \rightarrow \infty} y_n$,*

c) *Dacă $y_n \neq 0$ pentru orice $n \in \mathbb{N}$ și dacă $\lim_{n \rightarrow \infty} y_n \neq 0$, atunci şirul*

$$\left(\frac{x_n}{y_n} \right)_{n \in \mathbb{N}}$$
 este convergent și $\lim_{n \rightarrow \infty} \frac{x_n}{y_n} = \frac{\lim_{n \rightarrow \infty} x_n}{\lim_{n \rightarrow \infty} y_n}$.

Observație. *Dacă $(x_n)_{n \in \mathbb{N}}$, $(y_n)_{n \in \mathbb{N}}$ sunt şiruri convergente, $x = \lim_{n \rightarrow \infty} x_n$,*

$y = \lim_{n \rightarrow \infty} y_n$, iar $\alpha, \beta \in \mathbb{R}$, atunci $\lim_{n \rightarrow \infty} (\alpha x_n + \beta y_n) = \alpha x + \beta y$.

II.2. Trecerea la limită în inegalități

Propoziția 4. *Dacă $\lim_{n \rightarrow \infty} x_n = x$ și $\exists y \in \mathbb{R}$ și $n_0 \in \mathbb{N}$ astfel încât $x_n \leq y$ pentru orice $n \geq n_0$, atunci $x \leq y$.*

Corolar. *Dacă $\lim_{n \rightarrow \infty} x_n = x$, $\lim_{n \rightarrow \infty} y_n = y$ și există $n_0 \in \mathbb{N}$ astfel încât $x_n \leq y_n$ pentru orice $n \geq n_0$, atunci $x \leq y$.*

Propoziția 5. Dacă $(x_n)_{n \in \mathbb{N}}$ și $(y_n)_{n \in \mathbb{N}}$ sunt două siruri convergente având aceeași limită și pentru sirul $(z_n)_{n \in \mathbb{N}}$ există $n_0 \in \mathbb{N}$ astfel încât $x_n \leq z_n \leq y_n \quad \forall n \geq n_0$, atunci sirul $(z_n)_{n \in \mathbb{N}}$ este convergent și $\lim_{n \rightarrow \infty} z_n = \lim_{n \rightarrow \infty} x_n$.

Corolar. Fie sirurile $(x_n)_{n \in \mathbb{N}}$, $(y_n)_{n \in \mathbb{N}}$ astfel încât $\lim_{n \rightarrow \infty} y_n = 0$ și există $n_0 \in \mathbb{N}$ astfel încât $|x_n - x| \leq y_n, \quad \forall n \geq n_0$. Atunci $(x_n)_{n \in \mathbb{N}}$ este convergent și $\lim_{n \rightarrow \infty} x_n = x$.

II.3. Criterii de convergență

Definiție. Sirul $(x_n)_{n \in \mathbb{N}}$ se numește **crescător** (**descrescător**) dacă $x_n \leq x_{n+1}$ ($x_{n+1} \leq x_n$), $\forall n \in \mathbb{N}$.

Sirul $(x_n)_{n \in \mathbb{N}}$ se numește **monoton** dacă este crescător sau descrescător.

Teorema 2. Orice sir monoton și mărginit este convergent.

Definiție. Se spune că sirul $(x_n)_{n \in \mathbb{N}}$ are **limită** ∞ ($-\infty$) dacă pentru orice $M > 0$, există $n_0 \in \mathbb{N}$ astfel încât, pentru orice $n \geq n_0$ avem $x_n > M$ ($x_n < -M$). Se notează $\lim_{n \rightarrow \infty} x_n = \infty$ ($-\infty$).

Se spune că sirul $(x_n)_{n \in \mathbb{N}}$ are limită dacă este convergent, are limită $+\infty$ sau $-\infty$. Scriem pe scurt $\lim_{n \rightarrow \infty} x_n \in \bar{\mathbb{R}}$.

Teorema care urmează este folosită de obicei pentru tratarea, înlăturarea unor nedeterminări de forma $\frac{\infty}{\infty}$.

Teoremă 3 (Stolz). Fie $(a_n)_{n \in \mathbb{N}}$, $(b_n)_{n \in \mathbb{N}}$ două siruri de numere reale, $(b_n)_{n \in \mathbb{N}}$ strict crescător, nemărginit, $b_n > 0, \quad \forall n \in \mathbb{N}$ (sau $(b_n)_{n \in \mathbb{N}}$ strict descrescător, nemărginit, $b_n < 0, \quad \forall n \in \mathbb{N}$). Dacă sirul $\left(\frac{a_{n+1} - a_n}{b_{n+1} - b_n} \right)_{n \in \mathbb{N}}$ are limită,

atunci sirul $\left(\frac{a_n}{b_n} \right)_{n \in \mathbb{N}}$ are limită și $\lim_{n \rightarrow \infty} \frac{a_n}{b_n} = \lim_{n \rightarrow \infty} \frac{a_{n+1} - a_n}{b_{n+1} - b_n}$.

Corolar. Dacă $(x_n)_{n \in \mathbb{N}}$ are limită, atunci sirul $\left(\frac{x_1 + \dots + x_n}{n} \right)_{n \in \mathbb{N}^*}$ are limită și $\lim_{n \rightarrow \infty} \frac{x_1 + \dots + x_n}{n} = \lim_{n \rightarrow \infty} x_n$.

Fie $(x_n)_{n \in \mathbb{N}}$ un sir de numere reale si $(n_k)_{k \in \mathbb{N}}$ este un sir strict crescator de numere naturale. Sirul $(x_{n_k})_{k \in \mathbb{N}}$ se numeste sub sir al sirului $(x_n)_{n \in \mathbb{N}}$.

Observatie. Orice sub sir al unui sir convergent este convergent si are aceeași limită.

Lemă (Cesàro). Pentru orice sir mărginit există un sub sir convergent.

Definiție. Sirul $(x_n)_{n \in \mathbb{N}}$ se numește **sir Cauchy** (sau **sir fundamental**) dacă pentru orice $\varepsilon > 0$ există $n_0 \in \mathbb{N}$ astfel încât, pentru orice $n, m \geq n_0$ avem $|x_n - x_m| < \varepsilon$.

Propoziția 6. Orice sir Cauchy este mărginit.

Teorema 4 (criteriul lui Cauchy). Un sir de numere reale este convergent dacă și numai dacă este sir Cauchy.

Fie $(x_n)_{n \in \mathbb{N}}$ un sir mărginit. Notăm

$$\inf_{n \in \mathbb{N}} \sup_{k \geq n} x_n = \overline{\lim}_{n \in \mathbb{N}} x_n, \quad \sup_{n \in \mathbb{N}} \inf_{k \geq n} x_n = \underline{\lim}_{n \in \mathbb{N}} x_n$$

și le numim **limita superioară (inferioară)** a sirului.

Observatie. $\overline{\lim}_{n \in \mathbb{N}} x_n = \lim_{n \rightarrow \infty} y_n$, unde $y_n = \sup \{x_k \mid k \geq n\}$, iar

$$\underline{\lim}_{n \in \mathbb{N}} x_n = \lim_{n \rightarrow \infty} z_n, \text{ unde } z_n = \inf \{x_k \mid k \geq n\}.$$

Este evident că $\underline{\lim}_{n \in \mathbb{N}} x_n \leq \overline{\lim}_{n \in \mathbb{N}} x_n$.

Se poate arăta că:

Teorema 5. Sirul mărginit $(x_n)_{n \in \mathbb{N}}$ este convergent dacă și numai dacă $\underline{\lim}_{n \in \mathbb{N}} x_n = \overline{\lim}_{n \in \mathbb{N}} x_n$.

Teorema 6 (Teoplitz). Fie $(t_{nk})_{n \in \mathbb{N}^*, 1 \leq k \leq n}$ o matrice triunghiulară infinită cu proprietățile

i) există $M > 0$ astfel încât $\sum_{k=1}^n |t_{nk}| \leq M$, orice $n \in \mathbb{N}^*$,

ii) orice $n \in \mathbb{N}^*$, $\sum_{k=1}^n t_{nk} = 1$,

iii) orice $k \in \mathbb{N}^*$, $\lim_{n \rightarrow \infty} t_{nk} = 0$.

Atunci, pentru orice sir convergent $(x_n)_{n \in \mathbb{N}^*}$, sirul $(y_n)_{n \in \mathbb{N}^*}$ definit prin

$$y_n = \sum_{k=1}^n t_{nk} x_k \text{ este convergent și } \lim_{n \rightarrow \infty} y_n = \lim_{n \rightarrow \infty} x_n.$$

Exemple.

1. Fie $t_{nk} = \frac{1}{n}$, orice $k \in \{1, \dots, n\}$, orice $n \in \mathbb{N}^*$. Matricea $(t_{nk})_{\substack{n \in \mathbb{N}^* \\ 1 \leq k \leq n}}$ verifică ipotezele din teorema lui Toeplitz. Atunci pentru orice sir convergent $(x_n)_{n \geq 1}$ avem $\lim_{n \rightarrow \infty} \frac{x_1 + x_2 + \dots + x_n}{n} = \lim_{n \rightarrow \infty} x_n$, rezultat care s-a boținut și ca o consecință a teoremei lui Stolz.
2. Să se calculeze $\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{C_n^k}{2^n - 1} \left(1 + \frac{1}{k}\right)^k$. Se observă că sirul $y_n = \sum_{k=1}^n \frac{C_n^k}{2^n} \left(1 + \frac{1}{k}\right)^k$ este construit ca în teorema lui Toeplitz pornind de la matricea $\left(\frac{C_n^k}{2^n}\right)_{\substack{n \in \mathbb{N}^* \\ 1 \leq k \leq n}}$ și sirul $(x_k)_{k \in \mathbb{N}^*}$, $x_k = \left(1 + \frac{1}{k}\right)^k$. Conform teoremei amintite avem $\lim_{n \rightarrow \infty} y_n = \lim_{n \rightarrow \infty} x_n = e$.

III. SERII DE NUMERE REALE

III.1. Serii convergente

Definiție. Se numește **serie de numere reale** orice pereche $((x_n)_{n \in \mathbb{N}}, (s_n)_{n \in \mathbb{N}})$ de siruri de numere reale, unde $s_n = \sum_{k=0}^n x_k$. Pentru seria din definiție se folosește notația $\sum_{n \geq 0} x_n$. La fel ca la siruri, notația $\sum_{n \geq k} x_n$ are o interpretare naturală.

Se spune că x_n este termenul general (sau termenul de rang n) al seriei, iar s_n se numește suma parțială de ordinul n .

Definiție. Seria $\sum_{n \geq 0} x_n$ se numește **convergentă** dacă sirul $(s_n)_{n \in \mathbb{N}}$ este convergent. Se notează $\lim_{n \rightarrow \infty} s_n = \sum_{n=0}^{\infty} x_n$ și se spune că $\sum_{n=0}^{\infty} x_n$ este **suma seriei considerate**.

Dacă sirul $(s_n)_{n \in \mathbb{N}}$ nu este convergent se spune că seria $\sum_{n \geq 0} x_n$ este **divergentă**.

Propoziția 1. Dacă seria $\sum_{n \geq 0} x_n$ este convergentă, atunci $\lim_{n \rightarrow \infty} x_n = 0$.

Propoziția 2 (criteriul lui Cauchy). Seria $\sum_{n \geq 0} x_n$ este convergentă dacă și numai dacă pentru orice $\varepsilon > 0$ există $n_0 \in \mathbb{N}$ astfel încât pentru orice $n \geq n_0$ și

$$\text{orice } p \in \mathbb{N} \text{ avem } \left| \sum_{k=n}^{n+p} x_k \right| < \varepsilon.$$

Este bine să se observe că $\lim_{n \rightarrow \infty} \frac{1}{n} = 0$, deci propoziția 1 exprimă doar o condiție necesară de convergență a unei serii.

Fie $\sum_{n \geq 0} x_n$ o serie convergentă. Folosind, spre exemplu, propoziția 2, se poate observa că seria $\sum_{k \geq n} x_k$ este, de asemenea, convergentă. Pentru fiecare $n \in \mathbb{N}$ se notează atunci $r_n = \sum_{k=n+1}^{\infty} x_k$ și se numește restul de ordin n al seriei considerate.

Propoziția 3. Dacă $\sum_{n \geq 0} x_n$ este o serie convergentă, atunci sirul $(r_n)_{n \in \mathbb{N}}$ al resturilor acestei serii este convergent către zero.

Propoziția 4. Fie $\sum_{n \geq 0} x_n$, $\sum_{n \geq 0} y_n$ două serii convergente și $\lambda \in \mathbb{R}$. Atunci

i) seria $\sum_{n \geq 0} \lambda x_n$ este convergentă și $\sum_{n=0}^{\infty} \lambda x_n = \lambda \sum_{n=0}^{\infty} x_n$,

ii) seria $\sum_{n \geq 0} (x_n + y_n)$ este convergentă și $\sum_{n=0}^{\infty} (x_n + y_n) = \sum_{n=0}^{\infty} x_n + \sum_{n=0}^{\infty} y_n$.

III.2. Serii absolut convergente

Seria $\sum_{n \geq 0} x_n$ se numește absolut convergentă dacă seria $\sum_{n \geq 0} |x_n|$ este convergentă.

Propoziția 5. Dacă seria $\sum_{n \geq 0} x_n$ este absolut convergentă, atunci ea este convergentă și are loc $\left| \sum_{n=0}^{\infty} x_n \right| \leq \sum_{n=0}^{\infty} |x_n|$.

Propoziția 6 (criteriul lui Weierstrass). Dacă seria $\sum_{n \geq 0} y_n$ este convergentă și există $n_0 \in \mathbb{N}$ astfel încât $|x_n| \leq y_n$, $\forall n \geq n_0$, atunci seria $\sum_{n \geq 0} x_n$ este absolut convergentă.

Propoziția 7 (criteriul lui d'Alembert). Fie $\sum_{n \geq 0} x_n$ o serie cu termenii nenuli.

- i) Dacă $\overline{\lim}_{n \rightarrow \infty} \left| \frac{x_{n+1}}{x_n} \right| < 1$, atunci seria $\sum_{n \geq 0} x_n$ este absolut convergentă.
- ii) Dacă există $n_0 \in \mathbb{N}$ astfel încât $\left| \frac{x_{n+1}}{x_n} \right| \geq 1$ pentru orice $n \geq n_0$, atunci seria $\sum_{n \geq 0} x_n$ este divergentă.
- iii) Există atât serii divergente, cât și serii absolut convergente astfel încât $\underline{\lim}_{n \rightarrow \infty} \left| \frac{x_{n+1}}{x_n} \right| \leq 1 \leq \overline{\lim}_{n \rightarrow \infty} \left| \frac{x_{n+1}}{x_n} \right|$.

Observații. Criteriul se folosește de obicei în cazul în care există

$$\lim_{n \rightarrow \infty} \left| \frac{x_{n+1}}{x_n} \right|, \text{ deci:}$$

- i) Dacă $\lim_{n \rightarrow \infty} \left| \frac{x_{n+1}}{x_n} \right| < 1$, atunci seria $\sum_{n \geq 0} x_n$ este absolut convergentă;
- ii) Dacă $\lim_{n \rightarrow \infty} \left| \frac{x_{n+1}}{x_n} \right| > 1$, atunci seria $\sum_{n \geq 0} x_n$ este divergentă;
- iii) Există atât serii divergente, cât și serii absolut convergente astfel încât $\lim_{n \rightarrow \infty} \left| \frac{x_{n+1}}{x_n} \right| = 1$.

Propoziția 8 (criteriul radicalului). Fie $\sum_{n \geq 0} x_n$ o serie de numere reale.

- i) Dacă $\overline{\lim}^n \sqrt[n]{|x_n|} < 1$, atunci seria este absolut convergentă.
- ii) Dacă $\overline{\lim}^n \sqrt[n]{|x_n|} > 1$, atunci seria este divergentă.
- iii) Există atât serii divergente, cât și serii absolut convergente astfel încât $\overline{\lim}^n \sqrt[n]{|x_n|} = 1$.

Observație. Criteriul se folosește de obicei în cazul în care există $\lim_{n \rightarrow \infty} \sqrt[n]{|x_n|}$, deci:

- i) Dacă $\lim_{n \rightarrow \infty} \sqrt[n]{|x_n|} < 1$, atunci seria este divergentă.
- ii) Dacă $\lim_{n \rightarrow \infty} \sqrt[n]{|x_n|} > 1$, atunci seria este divergentă.
- iii) Există atât serii divergente, cât și serii absolut convergente astfel încât $\lim_{n \rightarrow \infty} \sqrt[n]{|x_n|} = 1$.

Propoziția 9 (criteriul Raabe – Duhamel). Fie $\sum_{n \geq 0} x_n$ o serie cu termeni nenuli.

- i) Dacă $\lim_{n \rightarrow \infty} n \left(\left| \frac{x_n}{x_{n+1}} \right| - 1 \right) > 1$, atunci seria este absolut convergentă.
- ii) Dacă $\lim_{n \rightarrow \infty} n \left(\left| \frac{x_n}{x_{n+1}} \right| - 1 \right) < 1$, atunci seria este divergentă.
- iii) Există atât serii divergente, cât și serii absolut convergente astfel încât $\lim_{n \rightarrow \infty} n \left(\left| \frac{x_n}{x_{n+1}} \right| - 1 \right) = 1$.

III.3. Serii cu termenii pozitivi

Este util să remarcăm că o serie cu termenii pozitivi este convergentă dacă și numai dacă sirul sumelor parțiale este mărginit.

Propoziția 10 (criteriul de condensare). Fie $\sum_{n \geq 0} x_n$ o serie cu termenii pozitivi astfel încât sirul $(x_n)_{n \in \mathbb{N}}$ este descrescător. Seria $\sum_{n \geq 0} x_n$ este convergentă dacă și numai dacă seria $\sum_{n \geq 0} 2^n x_n$ este convergentă.

Propoziția 11. Fie $\sum_{n \geq 0} x_n$, $\sum_{n \geq 0} y_n$ serii cu termenii strict pozitivi.

- i) Dacă sirul $\left(\frac{x_n}{y_n}\right)_{n \in \mathbb{N}}$ este convergent, iar seria $\sum_{n \geq 0} y_n$ este convergentă, atunci seria $\sum_{n \geq 0} x_n$ este convergentă.
- ii) Dacă $\lim_{n \rightarrow \infty} \frac{x_n}{y_n} \in (0, \infty]$ și dacă seria $\sum_{n \geq 0} y_n$ este divergentă, atunci seria $\sum_{n \geq 0} x_n$ este divergentă.

III.4. Serii cu termeni oarecari

Lemă (identitatea lui Abel). Fie $(x_n)_{n \in \mathbb{N}}$, $(y_n)_{n \in \mathbb{N}}$ două siruri de numere reale și $s_n = \sum_{k=0}^n x_k$. Are loc

$$\sum_{k=0}^n x_k y_k = s_n y_n + \sum_{k=0}^{n-1} s_k (y_k - y_{k+1}). \quad (1)$$

Propoziția 12 (criteriul lui Dirichlet). Fie $\sum_{n \geq 0} x_n$ o serie în care sirul sumelor parțiale este mărginit și fie $(y_n)_{n \in \mathbb{N}}$ un sir descrescător și convergent către zero. Atunci seria $\sum_{n \geq 0} x_n y_n$ este convergentă.

Propoziția 13 (criteriul lui Abel). Fie $\sum_{n \geq 0} x_n$ o serie convergentă și $(a_n)_{n \in \mathbb{N}}$ un sir monoton și mărginit. Atunci seria $\sum_{n \geq 0} a_n x_n$ este o serie convergentă.

Propoziția 14 (criteriul lui Leibniz). Fie $(x_n)_{n \in \mathbb{N}}$ un sir descrescător și convergent către zero. Atunci seria $\sum_{n \geq 0} (-1)^n x_n$ este convergentă și

$$\left| \sum_{k=0}^{\infty} (-1)^k x_k - \sum_{k=0}^n (-1)^k x_k \right| \leq x_{n+1}, \quad n \in \mathbb{N}.$$

III.5. Produsul a două serii

Definiție. Fie $\sum_{n \geq 0} x_n$, $\sum_{n \geq 0} y_n$ două serii de numere reale. Seria

$$\sum_{n \geq 0} \sum_{k=0}^n x_k y_{n-k} \quad se \quad numește \quad \text{produsul \ seriilor} \quad considerate \quad și \quad se \quad notează \\ \sum_{n \geq 0} x_n \times \sum_{n \geq 0} y_n .$$

Teorema 1 (Mertens). Dacă seriile $\sum_{n \geq 0} x_n$, $\sum_{n \geq 0} y_n$ sunt convergente și cel puțin una este absolut convergentă, atunci seria produs este convergentă și

$$\sum_{n=0}^{\infty} \sum_{k=0}^n x_k y_{n-k} = \left(\sum_{n=0}^{\infty} x_n \right) \left(\sum_{n=0}^{\infty} y_n \right).$$

Corolar. Dacă seriile $\sum_{n \geq 0} x_n$, $\sum_{n \geq 0} y_n$ sunt absolut convergente, atunci

seria produs este absolut convergentă și

$$\sum_{n=0}^{\infty} \left| \sum_{k=0}^n x_k y_{n-k} \right| = \left(\sum_{n=0}^{\infty} |x_n| \right) \left(\sum_{n=0}^{\infty} |y_n| \right).$$

III.6. Sumarea prin grupare

Fie $(n_k)_{k \in \mathbb{N}}$ un sir strict crescător de numere naturale, $\sum_{n \geq 0} x_n$ o serie

numerică și $y_k = \sum_{j=n_k+1}^{n_{k+1}} x_j$, $k \in \mathbb{N}$.

Spunem că seria $\sum_{k \geq 0} y_k$ s-a obținut prin gruparea termenilor seriei $\sum_{n \geq 0} x_n$.

Propoziția 15. Dacă seria $\sum_{n \geq 0} x_n$ este convergentă, atunci seria $\sum_{k \geq 0} y_k$,

obținută prin grupare, este convergentă și $\sum_{n=0}^{\infty} x_n = \sum_{n=0}^{\infty} y_n$.

Dacă seria $\sum_{n \geq 0} x_n$ are termenii pozitivi, iar seria $\sum_{k \geq 0} y_k$ este convergentă,

atunci seria $\sum_{n \geq 0} x_n$ este convergentă.

Propoziția 16. Fie $\sum_{n \geq 0} x_n$ o serie numerică în care $\lim_{n \rightarrow \infty} x_n = 0$ și fie $(n_k)_{k \in \mathbb{N}}$ un sir strict crescător de numere naturale astfel încât $(n_{k+1} - n_k)_{k \in \mathbb{N}}$ este mărginit. Dacă seria obținută prin grupare este convergentă, atunci seria $\sum_{n \geq 0} x_n$ este convergentă.

III.7. Permutarea unei serii

Definiție. Fie $\sum_{n \geq 0} x_n$ o serie numerică și $f : \mathbb{N} \rightarrow \mathbb{N}$ o bijecție. Spunem că seria $\sum_{k \geq 0} x_{f(k)}$ este o **permutare** a seriei $\sum_{n \geq 0} x_n$.

Teorema 2 (Dirichlet). Fie $\sum_{n \geq 0} x_n$ o serie numerică. Afirmațiile următoare

sunt echivalente:

- i) Seria considerată este absolut convergentă.
- ii) Orice permutare este absolut convergentă.
- iii) Orice permutare este convergentă.

Suma seriei permute coincide atunci cu suma seriei inițiale.

IV. ELEMENTE DE TOPOLOGIE ÎN \mathbb{R}

Definiție. O mulțime $G \subset \mathbb{R}$ se numește **deschisă** dacă pentru orice $x \in G$ există I un interval deschis astfel încât $x \in I \subset G$.

Mulțimea vidă se consideră deschisă.

Notăm cu \mathcal{G} familia mulțimilor deschise din \mathbb{R} .

Propoziția 1. Familia \mathcal{G} are proprietățile:

- i) $\emptyset, \mathbb{R} \in \mathcal{G}$,
- ii) $G_1, G_2 \in \mathcal{G} \Rightarrow G_1 \cap G_2 \in \mathcal{G}$,
- iii) Dacă $(G_i)_{i \in I} \subset \mathcal{G}$, atunci $\bigcup_{i \in I} G_i \in \mathcal{G}$.

Observație. O mulțime este deschisă dacă și numai dacă este o reuniune de intervale deschise. În particular, orice interval deschis este o mulțime deschisă.

Se poate arăta că o mulțime deschisă se poate scrie în mod unic ca o reuniune numărabilă de intervale deschise, disjuncte două câte două.

Amintim că o mulțime $V \subset \mathbb{R}$ se numește vecinătate pentru punctul $x \in \mathbb{R}$ dacă există $\varepsilon > 0$ astfel încât $(x - \varepsilon, x + \varepsilon) \subset V$.

Se observă atunci că mulțimea V este vecinătate pentru punctul x dacă există o mulțime deschisă G astfel încât $x \in G \subset V$.

Definiție. Punctul $x \in \mathbb{R}$ se numește **interior** mulțimii $A \subset \mathbb{R}$ dacă există un interval deschis I astfel încât $x \in I \subset A$.

Se notează $\overset{\circ}{A}$ sau $\text{int } A$ mulțimea punctelor interioare lui A și se numește interiorul lui A .

Este evident că $\overset{\circ}{A} \subset A$.

Propoziție 2. Interiorul unei mulțimi este o mulțime deschisă și este cea mai mare mulțime deschisă inclusă în acea mulțime. Mulțimea A este deschisă dacă și numai dacă $A = \overset{\circ}{A}$.

Dacă A este finită, atunci $\text{int } A = \emptyset$. Dacă $A = [a, b]$, atunci $\text{int } A = (a, b)$.

Definiție. Mulțimea $F \subset \mathbb{R}$ se numește **închisă** dacă complementara ei este o mulțime deschisă.

Propoziție 3. Mulțimile închise au următoarele proprietăți:

- i) \mathbb{R} și \emptyset sunt mulțimi închise;
- ii) Dacă F_1 și F_2 sunt mulțimi închise, atunci $F_1 \cup F_2$ este închisă;
- iii) Dacă $(F_i)_{i \in I}$ este o familie de mulțimi închise, atunci $\bigcap_{i \in I} F_i$ este închisă.

Definiție. Punctul $x \in \mathbb{R}$ se numește **aderent** mulțimii $A \subset \mathbb{R}$ dacă pentru orice vecinătate V a lui x avem $V \cap A \neq \emptyset$.

Se notează $\bar{A} = \{x \in \mathbb{R} \mid x \text{ aderent mulțimii } A\}$ și se numește aderența sau închiderea lui A .

Este evident că $A \subset \bar{A}$.

Orice punct de acumulare este punct aderent.

Propoziție 4. Aderența unei mulțimi este o mulțime închisă și este cea mai mică mulțime închisă care conține acea mulțime. Mulțimea A este închisă dacă și numai dacă $A = \bar{A}$.

Propoziție 5. Dacă $A \subset \mathbb{R}$, atunci $C\overset{\circ}{A} = \overline{CA}$ și $C\bar{A} = \overset{\circ}{CA}$.

Propoziție 6. Mulțimea A este închisă dacă și numai dacă pentru orice sir $(a_n)_{n \in \mathbb{N}}$, $a_n \in A$, convergent, limita se află de asemenea în A .

Propoziție 7. Dacă $A \subset \mathbb{R}$, atunci $a \in \bar{A}$ dacă și numai dacă există un sir $(a_n)_{n \in \mathbb{N}}$, $a_n \in A$, astfel încât $\lim_{n \rightarrow \infty} a_n = a$.

Definiție. Numărul x se numește **punct frontieră** pentru mulțimea $A \subset \mathbb{R}$ dacă este aderent atât mulțimii A cât și complementarei ei.

Mulțimea punctelor frontieră se notează ∂A (sau $\text{fr } A$) și se numește **frontiera** lui A ,

$$\partial A = \overline{A} \cap \overline{\mathbf{C}A}$$

Exemple.

1. $A = \left\{1, \frac{1}{2}, \dots, \frac{1}{n}, \dots\right\}$, atunci $\partial A = \left\{0, 1, \frac{1}{2}, \dots, \frac{1}{n}, \dots\right\}$.
2. $\partial \mathbb{Q} = \mathbb{R}$.
3. $\partial [0,1] = \{0,1\}$.

Definiție. Punctul $x \in A \subset \mathbb{R}$ se numește **izolat** pentru mulțimea A dacă există V o vecinătate a lui x astfel încât $A \cap V = \{x\}$.

Exemple.

1. Toate punctele lui \mathbb{N} (\mathbb{Z}) sunt izolate.
2. Mulțimea \mathbb{Q} nu are puncte izolate.
3. Un punct al mulțimii A care nu este punct de acumulare pentru A este punct izolat al lui A .

Definiție. O submulțime a lui \mathbb{R} se numește **compactă** dacă este mărginită și închisă.

Teoremă. O mulțime $A \subset \mathbb{R}$ este compactă dacă și numai dacă pentru orice sir $(a_n)_{n \in \mathbb{N}}$ cu elemente din A , există $(a_{n_k})_{k \in \mathbb{N}}$ un subșir convergent către un punct din A .

Definiție. Mulțimile $A, B \subset \mathbb{R}$ se numesc **separate** dacă $\overline{A} \cap B = A \cap \overline{B} = \emptyset$.

Mulțimi conexe

Definiție. O mulțime $A \subset \mathbb{R}$ se numește **conexă** dacă nu este reuniunea a două mulțimi nevide și separate. În caz contrar se spune că mulțimea A este **neconexă**.

Se poate demonstra:

Teoremă. Dacă $(A_i)_{i \in I}$ este o familie de mulțimi conexe, două câte două neseparate, atunci $\bigcup_{i \in I} A_i$ este conexă.

Teoremă. Dacă A este conexă și nevidă și $A \subset B \subset \overline{A}$, atunci B este conexă.

Teoremă. O submulțime din \mathbb{R} este conexă dacă și numai dacă este un interval.

V. FUNCȚII REALE CONTINUE

V.1. Limite de funcții

Fie funcția $f : E \rightarrow \mathbb{R}$ și x_0 un punct de acumulare al mulțimii E (x_0 finit sau infinit).

Definiție. Se spune că un număr l (finit sau infinit) este **limită** funcției f în punctul x_0 dacă pentru orice vecinătate U a lui l există o vecinătate V a lui x_0 astfel încât oricare ar fi $x \neq x_0$ din $V \cap E$ să avem $f(x) \in U$. Scriem $l = \lim_{\substack{x \rightarrow x_0 \\ x \neq x_0}} f(x)$ sau simplu, $l = \lim_{x \leftarrow x_0} f(x)$.

Teorema 1. Funcția f are limita l în punctul x_0 dacă și numai dacă pentru orice sir $(x_n)_{n \in \mathbb{N}}$, $x_n \in E$, $x_n \neq x_0$, $\lim_{n \rightarrow \infty} x_n = x_0$, sirul $(f(x_n))_{n \in \mathbb{N}}$ are limită și $\lim_{n \rightarrow \infty} f(x_n) = l$.

Definiție. Un număr l_s (l_d) (finit sau infinit) se numește **limite la stânga** (**limite la dreapta**) a funcției f în punctul x_0 dacă pentru orice vecinătate U a lui l_s (l_d) există o vecinătate V a lui x_0 astfel încât oricare ar fi $x < x_0$ ($x > x_0$) din $V \cap E$ să avem $f(x) \in U$. Scriem

$$l_s = \lim_{\substack{x \rightarrow x_0 \\ x < x_0}} f(x) = \lim_{x \uparrow x_0} f(x) = f(x_0 - 0)$$

$$(respectiv l_d = \lim_{\substack{x \rightarrow x_0 \\ x > x_0}} f(x) = \lim_{x \downarrow x_0} f(x) = f(x_0 + 0))$$

Propoziția 1. Un număr l_s (l_d) este limita la stânga (limita la dreapta) a funcției f în punctul x_0 dacă și numai dacă pentru orice sir $(x_n)_{n \in \mathbb{N}}$, $x_n \in E$, $x_n < x_0$, ($x_n > x_0$) $\lim_{n \rightarrow \infty} x_n = x_0$, sirul $(f(x_n))_{n \in \mathbb{N}}$ are limită și $\lim_{n \rightarrow \infty} f(x_n) = l_s$ ($\lim_{n \rightarrow \infty} f(x_n) = l_d$).

Propoziția 2. $l_s = \lim_{x \uparrow x_0} f(x)$ ($l_d = \lim_{x \downarrow x_0} f(x)$) dacă și numai dacă pentru orice sir crescător (descrescător) $(x_n)_{n \in \mathbb{N}}$, $x_n \in E$, $\lim_{n \rightarrow \infty} x_n = x_0$, sirul $(f(x_n))_{n \in \mathbb{N}}$ are limită și $\lim_{n \rightarrow \infty} f(x_n) = l_s$ ($\lim_{n \rightarrow \infty} f(x_n) = l_d$).

Propoziție 3. O funcție monotonă pe nulțimea E are limite laterale în orice punct de acumulare al mulțimii E .

Propoziția 4. Funcția f are limită în x_0 dacă și numai dacă are în x_0 limite laterale egale și avem

$$\lim_{x \rightarrow x_0} f(x) = f(x_0 - 0) = f(x_0 + 0).$$

Propoziția 5. Dacă limita funcției f în punctul x_0 există, atunci ea este unică.

Propoziția 6. Dacă funcția f are limită în x_0 , atunci funcția $|f|$ are limită în x_0 și

$$\lim_{x \rightarrow x_0} |f(x)| = \left| \lim_{x \rightarrow x_0} f(x) \right|.$$

Propoziția 7. Dacă l este finit, atunci

$$\lim_{x \rightarrow x_0} f(x) = l \Leftrightarrow \lim_{x \rightarrow x_0} [f(x) - l] = 0.$$

Teorema 2 (criteriul lui Cauchy-Bolzano). Funcția f are limită finită în punctul x_0 dacă și numai dacă pentru orice număr $\varepsilon > 0$ există o vecinătate V a lui x_0 astfel încât oricare ar fi punctele $x' \neq x_0$ și $x'' \neq x_0$ din $V \cap E$ să avem $|f(x') - f(x'')| < \varepsilon$.

Propoziția 8. Fie $f, g : E \rightarrow \mathbb{R}$.

Dacă $\lim_{x \rightarrow x_0} g(x) = 0$ și dacă există un număr finit l și o vecinătate V a lui x_0 astfel încât $|f(x) - l| \leq g(x)$ pentru orice $x \in V \cap E$, $x \neq x_0$, atunci $\lim_{x \rightarrow x_0} f(x) = l$.

Dacă $\lim_{x \rightarrow x_0} f(x) = +\infty$ ($\lim_{x \rightarrow x_0} f(x) = -\infty$) și dacă există o vecinătate V a lui x_0 astfel încât $f(x) \geq g(x)$ ($f(x) \leq g(x)$) pentru orice $x \in V \cap E$, $x \neq x_0$, atunci $\lim_{x \rightarrow x_0} f(x) = +\infty$ ($\lim_{x \rightarrow x_0} f(x) = -\infty$).

Teorema 3. Dacă funcțiile f și g au limite în punctul x_0 (finite sau infinite), atunci

1. dacă suma limitelor are sens, atunci funcția $f + g$ are limită în x_0 și

$$\lim_{x \rightarrow x_0} (f(x) + g(x)) = \lim_{x \rightarrow x_0} f(x) + \lim_{x \rightarrow x_0} g(x),$$

2. dacă produsul limitelor are sens, atunci funcția fg are limită în x_0 și

$$\lim_{x \rightarrow x_0} (f(x)g(x)) = \lim_{x \rightarrow x_0} f(x) \cdot \lim_{x \rightarrow x_0} g(x),$$

3. funcția αf are limită în x_0 , orice $\alpha \in R$ și

$$\lim_{x \rightarrow x_0} \alpha f(x) = \alpha \lim_{x \rightarrow x_0} f(x),$$

4. dacă raportul limitelor are sens, atunci funcția $\frac{f}{g}$ are limită în x_0 și

$$\lim_{x \rightarrow x_0} \frac{f(x)}{g(x)} = \frac{\lim_{x \rightarrow x_0} f(x)}{\lim_{x \rightarrow x_0} g(x)},$$

5. dacă $f \geq 0$ și dacă puterea limitelor lui f și g în x_0 are sens, atunci f^g are limită în x_0 și

$$\lim_{x \rightarrow x_0} (f(x)^{g(x)}) = \left[\lim_{x \rightarrow x_0} f(x) \right]^{\lim_{x \rightarrow x_0} g(x)}$$

Fie funcțiile $u : E \rightarrow F$, $f : F \rightarrow \mathbb{R}$ și funcția compusă $f \circ u : E \rightarrow \mathbb{R}$, $(f \circ u)(x) = f(u(x))$ pentru $x \in E$. Fie x_0 un punct de acumulare al lui E și u_0 un punct de acumulare al lui F .

Teorema 4. Dacă $\lim_{x \rightarrow x_0} u(x) = u_0$, $u(x) \neq u_0$ pentru $x \neq x_0$ și

$\lim_{u \rightarrow u_0} f(u) = l$, atunci funcția compusă $f \circ u$ are limită în x_0 și

$$\lim_{x \rightarrow x_0} (f \circ u)(x) = l.$$

Corolar 1. Dacă $\lim_{x \rightarrow x_0} u(x) = u_0$, $u_0 \in F$ și $\lim_{u \rightarrow u_0} f(u) = f(u_0)$, atunci

$$\lim_{x \rightarrow x_0} f(u(x)) = f\left(\lim_{x \rightarrow x_0} u(x)\right).$$

Corolar 2. Dacă $x_0 \in E$, $\lim_{x \rightarrow x_0} u(x) = u(x_0) = u_0$ și $\lim_{u \rightarrow u_0} f(u) = f(u_0)$, atunci $\lim_{x \rightarrow x_0} f(u(x)) = f(u(x_0))$.

V.2. Funcții continue

Definiție. Se spune că o funcție $f : E \rightarrow \mathbb{R}$ este continuă într-un punct $x_0 \in E$ dacă pentru orice vecinătate U a lui $f(x_0)$ există o vecinătate V a lui x_0 astfel încât oricare ar fi $x \in V \cap E$ să avem $f(x) \in U$.

Propoziția 1. Funcția $f : E \rightarrow \mathbb{R}$ este continuă într-un punct $x_0 \in E$ dacă și numai dacă pentru orice număr $\varepsilon > 0$ există un număr $\delta = \delta(\varepsilon) > 0$ astfel încât oricare ar fi $x \in E$ cu $|x - x_0| < \delta$ să avem $|f(x) - f(x_0)| < \varepsilon$.

Propoziția 2. O funcție $f:E \rightarrow \mathbb{R}$ este continuă într-un punct de acumulare $x_0 \in E$ dacă și numai dacă funcția are limită în x_0 și $\lim_{x \rightarrow x_0} f(x) = f(x_0)$.

Definiție. Se spune că o funcție $f:E \rightarrow \mathbb{R}$ este **continuă** pe o mulțime $A \subset E$ dacă este continuă în fiecare punct din A .

V.3. Proprietăți ale funcțiilor continue

Teorema 1. Dacă funcțiile $f, g:E \rightarrow \mathbb{R}$ sunt continue într-un punct $x_0 \in E$, atunci

1. $f+g$ este continuă în x_0 ,
2. αf este continuă în x_0 , oricare ar fi $\alpha \in \mathbb{R}$,
3. fg este continuă în x_0 ,
4. dacă $g(x_0) \neq 0$, funcția $\frac{f}{g}$ este continuă în x_0 ,
5. dacă $f(x_0)^{g(x_0)}$ are sens, atunci funcția f^g este continuă în x_0 .

Propoziția 1. Dacă funcția $f:E \rightarrow \mathbb{R}$ este continuă în $x_0 \in E$ (sau pe E), atunci funcția $|f|$ este continuă în x_0 (sau pe E).

Propoziția 2. Dacă funcțiile $f, g:E \rightarrow \mathbb{R}$ sunt continue în x_0 (sau pe E), atunci funcțiile $\sup(f, g)$ și $\inf(f, g)$ sunt continue în x_0 (sau pe E).

Fie o mulțime E , un punct de acumulare $x_0 \in E$ și o funcție $f:E \setminus \{x_0\} \rightarrow \mathbb{R}$.

Propoziția 3. Dacă funcția f are limită finită y_0 în punctul x_0 , atunci funcția $\bar{f}:E \rightarrow \mathbb{R}$, definită prin

$$\bar{f}(x) = \begin{cases} f(x) & \text{pentru } x \neq x_0 \\ y_0 & \text{pentru } x = x_0 \end{cases}$$

este continuă în x_0 . Funcția \bar{f} se numește **prelungirea prin continuitate a funcției f în punctul x_0** .

Fie $u:E \rightarrow F$ și $\varphi:F \rightarrow \mathbb{R}$.

Teorema 2. Dacă u este continuă în $x_0 \in E$ și φ este continuă în $u_u = u(x_0) \in F$, atunci funcția compusă $f = \varphi \circ u:E \rightarrow \mathbb{R}$ este continuă în x_0 .

Corolar. Dacă u este continuă pe E și φ este continuă pe F , atunci $f = \varphi \circ u$ este continuă pe E .

V.4. Proprietatea lui Darboux

Definiție. O funcție $f : I \rightarrow \mathbb{R}$, I interval, are proprietatea lui Darboux pe I dacă oricare ar fi punctele $a, b \in I$, $a < b$ și oricare ar fi numărul λ cuprins între $f(a)$ și $f(b)$ există un punct c_λ , $a < c_\lambda < b$ astfel încât $f(c_\lambda) = \lambda$.

Teorema 1. Dacă $f : I \rightarrow \mathbb{R}$ este continuă pe I , atunci f are proprietatea lui Darboux pe I .

Propoziție 1. Fie funcția $f : I \rightarrow \mathbb{R}$ și $a, b \in I$, $a < b$. Dacă f are proprietatea lui Darboux și dacă $f(a)f(b) < 0$, atunci există $c \in (a, b)$ astfel încât $f(c) = 0$.

Propoziția 2. O funcție $f : I \rightarrow \mathbb{R}$ are proprietatea lui Darboux dacă și numai dacă transformă orice interval $J \subset I$ tot într-un interval $f(J)$.

Corolar. Dacă $f : I \rightarrow \mathbb{R}$, I interval, are proprietatea lui Darboux, atunci $f(I)$ este interval.

Propoziția 3. Dacă $f : I \rightarrow \mathbb{R}$ are proprietatea lui Darboux și este bijectivă, atunci f este strict monotonă.

Propoziția 4. Dacă $f : I \rightarrow \mathbb{R}$ are proprietatea lui Darboux și dacă există una din limitele laterale în $x_0 \in I$, atunci ea este egală cu $f(x_0)$.

V.5. Continuitatea funcțiilor inverse

Propoziția 1. Dacă $f : E \rightarrow \mathbb{R}$ este monotonă și $f(E)$ este interval, atunci f este continuă pe E .

Teorema 1. Dacă $f : I \rightarrow J$, I, J intervale, este o aplicație strict monotonă, atunci f și f^{-1} sunt continue.

Teorema 2. O funcție continuă pe o mulțime compactă este mărginită pe această mulțime.

Teorema 3. Dacă f este continuă pe o mulțime compactă E , atunci $f(E)$ este compactă.

Teorema 4. O funcție continuă pe o mulțime compactă își atinge marginile pe această mulțime.

VI. ȘIRURI DE FUNCȚII CONTINUE

VI.1. Mulțimea de convergență

Fie A o mulțime oarecare și $f_1, f_2, \dots, f_n, \dots$ un sir de funcții reale definite toate pe A . Vom nota prescurtat acest sir de funcții prin $(f_n)_{n \in \mathbb{N}}$. Fie $a \in A$. Valorile funcțiilor din sirul $(f_n)_{n \in \mathbb{N}}$ în punctul a formează un sir de numere $(f_n(a))_{n \in \mathbb{N}}$. Așadar, pentru fiecare punct $x \in A$, putem considera sirul de numere $(f_n(x))_{n \in \mathbb{N}}$ format cu valorile funcțiilor din sirul $(f_n)_{n \in \mathbb{N}}$ în punctul x .

Un sir de funcții $(f_n)_{n \in \mathbb{N}}$ este deci echivalent cu o familie de siruri de numere $(f_n(x))_{n \in \mathbb{N}, x \in A}$.

Vom spune că un punct a este un **punct de convergență** al sirului de funcții $(f_n)_{n \in \mathbb{N}}$ dacă sirul de numere $(f_n(a))_{n \in \mathbb{N}}$ este convergent. Mulțimea punctelor de convergență ale sirului de funcții $(f_n)_{n \in \mathbb{N}}$ se numește **mulțimea de convergență** a sirului $(f_n)_{n \in \mathbb{N}}$.

Fie B mulțimea de convergență a sirului $(f_n)_{n \in \mathbb{N}}$. Pentru fiecare punct $x \in B$ să notăm cu $f(x)$ limita sirului de numere $(f_n(x))_{n \in \mathbb{N}}$, $\lim_{n \rightarrow \infty} f_n(x) = f(x)$. Am stabilit astfel o corespondență între punctele $x \in B$ și numerele reale, adică o funcție $f : B \rightarrow \mathbb{R}$ definită prin $f(x) = \lim_{n \rightarrow \infty} f_n(x)$.

Funcția f se numește **funcția limită**, pe mulțimea B , a sirului de funcții $(f_n)_{n \in \mathbb{N}}$. Vom spune că sirul $(f_n)_{n \in \mathbb{N}}$ converge pe mulțimea B către funcția f .

VI.2. Convergență simplă și convergență uniformă

Fie $(f_n)_{n \in \mathbb{N}}$ un sir de funcții definite pe o mulțime A . Vom spune că o funcție $f : A \rightarrow \mathbb{R}$ este limita simplă (sau punctuală) a sirului $(f_n)_{n \in \mathbb{N}}$, sau că sirul $(f_n)_{n \in \mathbb{N}}$ converge simplu (sau punctual) pe A către f , dacă pentru fiecare punct $x \in A$, sirul de numere $(f_n(x))_{n \in \mathbb{N}}$ este convergent către numărul $f(x)$. Cu alte cuvinte:

Definiție. *Sirul $(f_n)_{n \in \mathbb{N}}$ este simplu convergent pe A către f dacă oricare ar fi $x \in A$ și oricare ar fi $\varepsilon > 0$ există un număr $N(\varepsilon, x)$ astfel încât pentru orice $n \geq N(\varepsilon, x)$ să avem $|f_n(x) - f(x)| < \varepsilon$. Vom scrie $f_n \xrightarrow{s} f$.*

Dacă în definiția precedentă numărul N depinde numai de ε , nu și de x vom spune că sirul $(f_n)_{n \in \mathbb{N}}$ este uniform convergent pe A către f . Astfel:

Definiție. *Sirul $(f_n)_{n \in \mathbb{N}}$ este uniform convergent pe A către f dacă oricare ar fi $\varepsilon > 0$ există un număr $N(\varepsilon)$ astfel încât oricare ar fi $n \geq N(\varepsilon)$ și oricare ar fi $x \in A$ avem $|f_n(x) - f(x)| < \varepsilon$. Vom scrie $f_n \xrightarrow{u} f$.*

Evident, dacă sirul $(f_n)_{n \in \mathbb{N}}$ este uniform convergent, el este simplu convergent pe A către f . Afirmația reciprocă nu este, în general, adevărată.

Deoarece convergența simplă a sirului de funcții $(f_n)_{n \in \mathbb{N}}$ către funcția f revine, pentru fiecare $x \in A$, la convergența sirului de numere $(f_n(x))_{n \in \mathbb{N}}$ către numărul $f(x)$, criteriile de convergență de la sirurile de numere se aplică și în acest caz.

Teorema 1 (criteriul lui Cauchy). *Fie $(f_n)_{n \in \mathbb{N}}$ un sir de funcții definite pe o mulțime A . Sirul $(f_n)_{n \in \mathbb{N}}$ este uniform convergent către o funcție $f : A \rightarrow \mathbb{R}$ dacă și numai dacă pentru orice $\varepsilon > 0$ există un număr $N(\varepsilon)$ astfel încât oricare ar fi $n, m \geq N(\varepsilon)$ și oricare ar fi $x \in A$ avem $|f_n(x) - f_m(x)| < \varepsilon$.*

Teorema 2. *Fie $(f_n)_{n \in \mathbb{N}}$ și $(\varphi_n)_{n \in \mathbb{N}}$ două siruri de funcții definite pe A și $f : A \rightarrow \mathbb{R}$. Dacă $|f_n(x) - f(x)| \leq \varphi_n(x)$ pentru orice $n \in \mathbb{N}$ și orice $x \in A$ și dacă $\varphi_n \xrightarrow{u} 0$, atunci $f_n \xrightarrow{u} f$.*

Corolar. *Fie $(f_n)_{n \in \mathbb{N}}$ un sir de funcții definit pe A și $f : A \rightarrow \mathbb{R}$. Dacă există un sir de numere $(a_n)_{n \in \mathbb{N}}$ astfel încât să avem $|f_n(x) - f(x)| \leq a_n$ pentru orice $n \in \mathbb{N}$ și orice $x \in A$ și dacă $a_n \rightarrow 0$, atunci $f_n \xrightarrow{u} f$.*

Se ridică în mod natural întrebarea dacă o anumită proprietate pe care o au toate funcțiile unui sir $(f_n)_{n \in \mathbb{N}}$.

Teorema 3. *Fie $(f_n)_{n \in \mathbb{N}}$ un sir uniform convergent pe mulțimea $A \subset \mathbb{R}$ către funcția f . Dacă toate funcțiile f_n sunt continue într-un punct $a \in A$, atunci funcția limită f este continuă în punctul a .*

Corolar. Limita unui sir convergent de functii continue pe A este o functie continua pe A .

Propozitie 1. Dacă $(f_n)_{n \in \mathbb{N}}$ este un sir de functii mărginite pe A și uniform convergent către o functie f , atunci functia limită f este mărginită pe A .

VI.3. Aproximarea uniformă a functiilor continue

Functiile continue pot fi aproximate uniform cu functii aparținând unei mulțimi mai restrâns de functii continue.

Reamintim că o mulțime \mathcal{A} de functii continue definite pe o mulțime A este o algebră dacă suma și produsul a două functii din \mathcal{A} aparțin lui \mathcal{A} și produsul unei functii din \mathcal{A} cu un număr aparține lui \mathcal{A} .

Teorema 1 (Weierstrass-Stone). Fie intervalul compact $I = [a, b]$ și \mathcal{A} o algebră de functii continue definite pe I . Dacă

1. funcția identic egală cu 1 pe I , $f(x) \equiv 1$, aparține lui \mathcal{A} ,
2. pentru orice puncte $x' \neq x''$ există o funcție $f \in \mathcal{A}$ astfel încât $f(x') \neq f(x'')$,

atunci orice funcție continuă pe I este limita uniformă a unui sir de functii din \mathcal{A} .

Observație. Teorema 1 rămâne valabilă dacă se înlocuiește I cu un spațiu compact oarecare.

Teorema 2 (Weierstrass). Orice funcție continuă pe un interval compact $I = [a, b]$ este limita uniformă pe I a unui sir de polinoame.

VII. FUNCȚII DERIVABILE

VII.1. Derivata

Definiție. Se spune că funcția $f : I \rightarrow \mathbb{R}$ este derivabilă într-un punct $x_0 \in I$ dacă raportul $\frac{f(x) - f(x_0)}{x - x_0}$ are limită finită în punctul x_0 . Limita se numește **derivata** funcției f în punctul x_0 și se notează cu $f'(x_0)$,

$$f'(x_0) = \lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0}.$$

Teorema 1. Dacă funcția $f : I \rightarrow \mathbb{R}$ este derivabilă în punctul $x_0 \in I$, atunci f este continuă în x_0 .

Definiție. Se spune că funcția $f : I \rightarrow \mathbb{R}$ este derivabilă la stânga (derivabilă la dreapta) în punctul $x_0 \in I$, dacă raportul $\frac{f(x) - f(x_0)}{x - x_0}$ are limită

la stânga (la dreapta) în x_0 . Această limită se numește derivata la stânga (derivata la dreapta) a funcției f în x_0 și se notează

$$f'_s(x_0) = \lim_{\substack{x \rightarrow x_0 \\ x < x_0}} \frac{f(x) - f(x_0)}{x - x_0} \quad (f'_d(x_0) = \lim_{\substack{x \rightarrow x_0 \\ x > x_0}} \frac{f(x) - f(x_0)}{x - x_0}).$$

Propoziție 1. O funcție $f : I \rightarrow \mathbb{R}$ are derivată într-un punct interior $x_0 \in I$ dacă și numai dacă are derivate laterale egale în x_0 . În acest caz avem $f'(x_0) = f'_s(x_0) = f'_d(x_0)$.

Definiție. Se spune că funcția $f : I \rightarrow \mathbb{R}$ este derivabilă pe o submulțime $A \subset I$ dacă are derivată în fiecare punct din A .

Propoziția 2. Dacă funcția $f : I \rightarrow \mathbb{R}$ este derivabilă pe I , atunci f este continuă pe I .

VII.2. Operații cu funcții derivabile

Propoziția 1. Dacă funcțiile $f, g : I \rightarrow \mathbb{R}$ sunt derivabile în $x_0 \in I$ (pe I), atunci funcția $f + g$ este derivabilă în x_0 (pe I) și

$$(f + g)'(x_0) = f'(x_0) + g'(x_0) \quad ((f + g)' = f' + g').$$

Propoziția 2. Dacă funcția $f : I \rightarrow \mathbb{R}$ este derivabilă în $x_0 \in I$ (pe I), atunci funcția cf este derivabilă în x_0 (pe I) și

$$(cf)'(x_0) = cf'(x_0) \quad ((cf)' = cf')$$

Propoziția 3. Dacă funcțiile $f, g : I \rightarrow \mathbb{R}$ sunt derivabile în $x_0 \in I$ (pe I), atunci funcția fg este derivabilă în x_0 (pe I) și

$$(fg)'(x_0) = f'(x_0)g(x_0) + f(x_0)g'(x_0) \quad ((fg)' = f'g + fg').$$

Propoziția 4. Dacă funcțiile $f, g : I \rightarrow \mathbb{R}$ sunt derivabile în $x_0 \in I$ (pe I), și dacă $g(x_0) \neq 0$ ($g(x) \neq 0, \forall x \in I$), atunci funcția $\frac{f}{g}$ este derivabilă în x_0 (pe I) și

$$\left(\frac{f}{g}\right)'(x_0) = \frac{f'(x_0)g(x_0) - f(x_0)g'(x_0)}{[g(x_0)]^2} \quad \left(\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}\right).$$

Fie $u : I \rightarrow J$ și $\varphi : J \rightarrow \mathbb{R}$, I, J intervale.

Teorema 1. Dacă funcția u este derivabilă în $x_0 \in I$ (pe I) și funcția φ este derivabilă în $y_0 = u(x_0) \in J$ (pe J), atunci funcția compusă $f = \varphi \circ u$ este derivabilă în x_0 (pe I) și

$$f'(x_0) = \varphi'(u(x_0)) \cdot u'(x_0) \quad (f' = (\varphi' \circ u) \cdot u').$$

Teorema 2. Dacă funcția bijectivă $f : I \rightarrow J$ este derivabilă în $x_0 \in I$ și dacă $f'(x_0) \neq 0$, atunci funcția sa inversă $f^{-1} : J \rightarrow I$ este derivabilă în $y_0 = f(x_0) \in J$ și

$$(f^{-1})'(y_0) = \frac{1}{f'(x_0)}.$$

VII.3. Proprietățile funcțiilor derivabile

Fie $f : I \rightarrow \mathbb{R}$ și $x_0 \in I$.

Definiție. Se spune că x_0 este un punct de **maxim (minim) relativ** (sau local) al funcției f dacă există o vecinătate V a lui x_0 astfel încât să avem $f(x) \leq f(x_0)$ ($f(x) \geq f(x_0)$) pentru orice $x \in V \cap I$.

Teorema 1 (Fermat). Dacă funcția f este derivabilă într-u punct de extrem din interiorul intervalului I , atunci $f'(x_0) = 0$.

Teorema 2 (Rolle). Fie $f : I \rightarrow \mathbb{R}$ o funcție și $a < b$ două puncte din I .

Dacă:

1. funcția f este continuă pe intervalul închis $[a, b]$,
2. funcția f este derivabilă pe intervalul deschis (a, b) ,
3. $f(a) = f(b)$,

atunci există $c \in (a, b)$ astfel încât $f'(c) = 0$.

Teorema 3 (Lagrange). Fie $f : I \rightarrow \mathbb{R}$ o funcție și $a < b$ două puncte din I . Dacă

1. funcția f este continuă pe intervalul închis $[a, b]$,
 2. funcția f este derivabilă pe intervalul deschis (a, b) ,
- atunci există $c \in (a, b)$ astfel încât să avem

$$f(b) - f(a) = (b - a)f'(c).$$

Propoziția 1. Dacă f are derivata nulă pe un interval I , atunci funcția f este constantă pe I .

Propoziția 2. Dacă f și g sunt două funcții derivabile pe un interval I și dacă erivatele lor sunt egale pe I , atunci diferența celor două funcții este constantă pe intervalul I .

Propoziția 3. Fie f o funcție derivabilă pe un interval I . Dacă f' este strict pozitivă (negativă) pe I , atunci f este strict crescătoare (descrescătoare) pe I .

Teorema 4 (Cauchy). Fie f și g două funcții definite pe un interval I și $a < b$ două puncte din I . Dacă

1. f și g sunt continue pe intervalul închis $[a,b]$,
2. f și g sunt derivabile pe intervalul deschis (a,b) ,
3. $g'(x) \neq 0$ pentru orice $x \in (a,b)$,

atunci $g(a) \neq g(b)$ și există un punct $c \in (a,b)$ astfel încât să avem

$$\frac{f(b)-f(a)}{g(b)-g(a)} = \frac{f'(c)}{g'(c)}.$$

Teorema 5 (Darboux). Dacă f este derivabilă pe I , atunci derivata f' are proprietatea lui Darboux pe I .

VII.4. Derivate de ordin superior. Formula lui Taylor

Fie $f : I \rightarrow \mathbb{R}$ și $x_0 \in I$. Presupunem că f este derivabilă pe o vecinătate V a lui x_0 .

Definiție. Dacă derivata f' este derivabilă în x_0 se spune că funcția f este de două ori derivabilă în punctul x_0 ; Derivata lui f' în x_0 se notează $f''(x_0)$ (sau $\frac{d^2 f(x_0)}{dx^2}$, $D^2 f(x_0)$) și se numește derivata a două (sau derivata de ordinul doi) a funcției f în punctul x_0 ,

$$f''(x_0) = \lim_{\substack{x \rightarrow x_0 \\ x \in V}} \frac{f'(x) - f'(x_0)}{x - x_0}.$$

Observație. Dacă funcția f este derivabilă numai în punctul x_0 (sau pe o mulțime care nu are pe x_0 ca punct de acumulare), nu se mai poate defini derivata a doua a funcției f în punctul x_0 .

Definiție. Dacă funcția f este derivabilă de $n-1$ ori pe o vecinătate V a lui x_0 și dacă derivata $f^{(n-1)}$ este derivabilă în x_0 se spune că funcția f este de n ori derivabilă în x_0 ; derivata lui $f^{(n-1)}$ în punctul x_0 se numește derivata de ordinul n a funcției f în punctul x_0 ,

$$f^{(n)}(x_0) = \lim_{\substack{x \rightarrow x_0 \\ x \in V}} \frac{f^{(n-1)}(x) - f^{(n-1)}(x_0)}{x - x_0}.$$

Propoziția 1. Fie funcțiile $f, g : I \rightarrow \mathbb{R}$ și $\alpha \in \mathbb{R}$. Dacă f și g sunt de n ori derivabile în $x_0 \in I$, atunci $f + g$, αf și fg sunt derivabile de n ori în x_0 ; dacă în plus $g(x_0) \neq 0$, atunci $\frac{f}{g}$ este derivabilă de n ori în x_0 .

Corolar. Fie funcțiile $f, g : I \rightarrow \mathbb{R}$ și $\alpha \in \mathbb{R}$. Dacă f și g sunt de n ori derivabile pe I , atunci $f + g$, αf și fg sunt derivabile de n ori pe I și

$$(f + g)^{(n)} = f^{(n)} + g^{(n)}, (\alpha f)^{(n)} = \alpha f^{(n)}, (fg)^{(n)} = \sum_{i=1}^n C_n^i f^{(n-i)} g^{(i)};$$

$\frac{f}{g}$ este derivabilă de n ori pe mulțimea sa de definiție.

Fie funcția $f : I \rightarrow \mathbb{R}$, derivabilă de n ori într-un punct $a \in I$. Aceasta înseamnă că primele $n-1$ derivate există nu numai în a , dar pe o întreagă vecinătate a lui a . Pentru simplificare vom presupune că primele $n-1$ derivate există pe întregul interval I .

Pentru fiecare $x \in I$ să definim polinomul

$$T_n(x) = f(a) + \frac{x-a}{1!} f'(a) + \frac{(x-a)^2}{2!} f''(a) + \dots + \frac{(x-a)^n}{n!} f^{(n)}(a)$$

Polinomul T_n definit pe I se numește **polinomul lui Taylor** de gradul n atașat funcției f în punctul a .

Dacă pentru fiecare $x \in I$ notăm $R_n(x) = f(x) - T_n(x)$, atunci

$$T_n(x) = f(a) + \frac{x-a}{1!} f'(a) + \frac{(x-a)^2}{2!} f''(a) + \dots + \frac{(x-a)^n}{n!} f^{(n)}(a) + R_n(x)$$

oricare ar fi $x \in I$. Această egalitate, valabilă pentru orice $x \in I$ se numește **formula lui Taylor** de ordinul n corespunzătoare funcției f în punctul a . Funcția R_n definită pe I se numește **restul** de ordinul n al formulei lui Taylor.

Propoziție. Dacă f este derivabilă de n ori în punctul $a \in I$, atunci există o funcție $\alpha : I \rightarrow \mathbb{R}$ astfel ca $\lim_{x \rightarrow a} \alpha(x) = 0 = \alpha(a)$ și pentru orice $x \in I$ să avem

$$T_n(x) = f(a) + \frac{x-a}{1!} f'(a) + \frac{(x-a)^2}{2!} f''(a) + \dots + \frac{(x-a)^n}{n!} f^{(n)}(a) + \frac{(x-a)^n}{n!} \alpha(x).$$

VIII. FUNCȚII ANALITICE

VIII.1. Serii de puteri

Definiție. Se numește **serie de puteri** orice pereche $(a_n x^n)_{n \in \mathbb{N}}, (s_n)_{n \in \mathbb{N}}$

unde $(a_n)_{n \in \mathbb{N}}$ este un sir de numere reale iar $s_n = \sum_{k=0}^n a_k x^k$, $x \in \mathbb{R}$.

Perechea precedentă se notează $\sum_{n \geq 0} a_n x^n$.

Definiție. Se numește **rază de convergență** a seriei de puteri $\sum_{n \geq 0} a_n x^n$

numărul

$$\rho = \sup \left\{ r \geq 0 \mid \sum_{n \geq 0} |a_n| x^n \text{ este convergentă} \right\}.$$

Teorema 1 (Cauchy-Hadamard). Cu convențiile $\frac{1}{0} = \infty$, $\frac{1}{\infty} = 0$ are loc

$$\rho = \frac{1}{\limsup_{n \rightarrow \infty} \sqrt[n]{|a_n|}}$$

Definiție. Se spune că seria $\sum_{n \geq 0} a_n x^n$ este **uniform convergentă** pe mulțimea $A \subset \mathbb{R}$ dacă sirul $(s_n)_{n \in \mathbb{N}}$ al sumelor parțiale este uniform convergent pe A .

Propoziția 1. Dacă există $\sum_{n \geq 0} \varepsilon_n$ o serie convergentă și $A \subset \mathbb{R}$ astfel încât $|a_n x^n| \leq \varepsilon_n$ pentru orice $n \in \mathbb{N}$, $x \in A$, atunci seria $\sum_{n \geq 0} a_n x^n$ este uniform convergentă pe mulțimea A .

Teorema 2 (Abel). Fie $\sum_{n \geq 0} a_n x^n$ o serie de puteri cu raza de convergență $\rho > 0$. Atunci pentru orice $r \in (0, \rho)$ seria este absolut și uniform convergentă pe $[-r, r]$.

Funcția $f : (-\rho, \rho) \rightarrow \mathbb{R}$, $f(x) = \sum_{n=0}^{\infty} a_n x^n$ este indefinit derivabilă și

$$f^{(k)} = \sum_{n=k}^{\infty} n(n-1)\dots(n-k+1)a_n x^{n-k}, \quad k \geq 1.$$

Teorema 3 (Abel). Fie $\lambda \in \mathbb{R}$ astfel încât seria $\sum_{n \geq 0} a_n \lambda^n$ este convergentă. Atunci seria $\sum_{n \geq 0} a_n x^n$ este uniform convergentă pe $[0, \lambda]$.

VIII.2. Funcții transcendent elementare

Se notează

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}, \quad x \in \mathbb{R}.$$

Conform teoremei lui Abel funcția \exp este indefinit derivabilă și $\exp' = \exp$. Conform teoremei lui Martens avem $\exp(x+y) = \exp x \cdot \exp y$.

Din $\exp(0) = 1$, $\exp(1) = e$ rezultă $\exp r = e^r$ pentru orice număr rațional r .

Această proprietate impune notația $\exp x = e^x$, adică

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}, \quad x \in \mathbb{R}.$$

Vom spune că e^x este funcția exponențială în baza e .

Se arată că e^x este o bijecție strict crescătoare între \mathbb{R} și $(0, \infty)$, iar inversa acestei funcții se notează $\ln x$.

Dacă $a > 0$ și $x \in \mathbb{R}$, prin definiție $a^x = e^{x \ln a}$.

Se notează

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}, \quad \sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}.$$

VIII.3. Funcții analitice

Fiind dată o serie de puteri $\sum_{n \geq 0} a_n x^n$ cu raza de convergență $\rho \neq 0$, se consideră funcția $f: (-\rho, \rho) \rightarrow \mathbb{R}$, $f(x) = \sum_{n=0}^{\infty} a_n x^n$ care se va numi **suma seriei**.

Teoremă. Suma seriei $\sum_{n \geq 0} a_n x^n$ este egală cu 0 pe un interval nevid $(-\alpha, \alpha) \subset (-\rho, \rho)$ dacă și numai dacă toți coeficienții a_n sunt egali cu 0.

Corolar 1. Dacă suma seriei $\sum_{n \geq 0} a_n x^n$ nu este funcția nulă și dacă se anulează în 0, atunci 0 este punct izolat al mulțimii zerourilor acestei funcții.

Corolar 2. Dacă seriile de puteri $\sum_{n \geq 0} a_n x^n$, $\sum_{n \geq 0} b_n x^n$ au razele de convergență diferite de 0 și există $\alpha > 0$ astfel încât $\sum_{n \geq 0} a_n x^n = \sum_{n \geq 0} b_n x^n$ pentru orice $x \in (-\alpha, \alpha)$, atunci $a_n = b_n$ pentru orice $n \in \mathbb{N}$.

Fie $D \subset \mathbb{R}$ o mulțime deschisă.

Definiție. Funcția $f: D \rightarrow \mathbb{R}$ se numește **analitică** în punctul $x_0 \in D$ dacă există $r > 0$ și o serie de puteri $\sum_{n \geq 0} a_n x^n$ astfel încât $f(x) = \sum_{n=0}^{\infty} a_n (x - x_0)^n$ pentru orice $x \in (x_0 - r, x_0 + r)$.

Propoziție. Dacă funcția f este analitică în x_0 atunci ea este infinit derivabilă într-o vecinătate a punctului x_0 și $a_n = \frac{1}{n!} f^{(n)}(x_0)$, deci

$f(x) = \sum_{n=0}^{\infty} \frac{1}{n!} f^{(n)}(x - x_0)^n$ pentru orice $x \in (x_0 - r, x_0 + r)$.

Exemplu. Pentru $\alpha \in \mathbb{R}$, fie $f: (-1, +\infty) \rightarrow \mathbb{R}$, $f(x) = (1+x)^\alpha$. Atunci $(1+x)^\alpha = \sum_{n=0}^{\infty} \frac{\alpha(\alpha-1)\dots(\alpha-n+1)}{n!} x^n$, $x \in (-1, 1)$.

IX. INTEGRALA RIEMANN ÎN \mathbb{R}

IX. 1. Integrala Darboux

Fie $a, b \in \mathbb{R}$, $a < b$.

Definiție. O mulțime $\{x_0, \dots, x_n\}$ astfel încât $a = x_0 < x_1 < \dots < x_n = b$ se numește **diviziune** a intervalului $[a, b]$.

Vom nota cu Δ mulțimea tuturor diviziunilor intervalului $[a, b]$.

Pentru $\rho \in \Delta$, $\delta = \{x_0, \dots, x_n\}$ numărul

$$\|\delta\| = \max \{x_{i+1} - x_i \mid 0 \leq i \leq n-1\}$$

se va numi normă diviziunii δ .

Fie $f : [a, b] \rightarrow \mathbb{R}$ o funcție mărginită și $\rho \in \Delta$, $\delta = \{x_0, \dots, x_n\}$. Notăm

$$M_i = \sup \{f(x) \mid x \in [x_i, x_{i+1}]\}, \quad m_i = \inf \{f(x) \mid x \in [x_i, x_{i+1}]\},$$

$$S_\rho(f) = \sum_{i=0}^{n-1} M_i (x_{i+1} - x_i), \quad s_\rho(f) = \sum_{i=0}^{n-1} m_i (x_{i+1} - x_i).$$

Numerele $S_\rho(f)$, $s_\rho(f)$ se numesc **suma Darboux superioară** respectiv **inferioară** asociată diviziunii δ și funcției f .

Evident

$$(b-a) \inf_{x \in [a,b]} f(x) \leq s_\delta(f) \leq S_\delta(f) \leq \sup_{x \in [a,b]} f(x).$$

Se notează

$$\overline{\int} f(x) dx = \inf_{\rho \in \Delta} S_\rho(f), \quad \underline{\int} f(x) dx = \sup_{\rho \in \Delta} s_\rho(f)$$

care se numesc **integrala superioară** respectiv **inferioară** a funcției f pe intervalul $[a, b]$.

Definiție. Funcția mărginită $f : [a, b] \rightarrow \mathbb{R}$ se numește **integrabilă Riemann** pe intervalul $[a, b]$ dacă $\overline{\int} f(x) dx = \underline{\int} f(x) dx$. Se notează atunci

$$\int_a^b f(x) dx = \underline{\int} f(x) dx$$

și se numește **integrala Riemann** a funcției f pe intervalul $[a, b]$.

Se notează $\mathcal{R}[a, b]$ mulțimea funcțiilor integrabile Riemann pe intervalul $[a, b]$.

IX.2. Criterii de integrabilitate

Dacă $\delta_1, \delta_2 \in \Delta$ și $\delta_1 \subset \delta_2$ vom spune că diviziunea δ_2 este **mai fină** decât diviziunea δ_1 și vom nota $\delta_1 \leq \delta_2$.

Propoziția 1. Fie $f : [a, b] \rightarrow \mathbb{R}$ o funcție mărginită, $\delta_1, \delta_2 \in \Delta$, $\delta_1 \leq \delta_2$.

Atunci

$$s_{\delta_1}(f) \leq s_{\delta_2}(f) \leq S_{\delta_2}(f) \leq S_{\delta_1}(f).$$

Corolar 1. Dacă $\delta', \delta'' \in \Delta$, atunci $s_{\delta'}(f) \leq S_{\delta''}(f)$.

Corolar 2. Dacă $f : [a, b] \rightarrow \mathbb{R}$ este mărginită, atunci

$$\overline{\int} f(x) dx \geq \underline{\int} f(x) dx.$$

Teorema 1 (Criteriul lui Darboux). Funcția mărginită $f : [a, b] \rightarrow \mathbb{R}$ este integrabilă Riemann dacă și numai dacă pentru orice $\varepsilon > 0$ există $\delta \in \Delta$ astfel încât $S_\delta(f) - s_\delta(f) < \varepsilon$.

Corolar 1. Orice funcție continuă este integrabilă Riemann.

Corolar 2. Orice funcție monotonă este integrabilă Riemann.

Fie $\rho \in \Delta$, $\rho = \{x_0, \dots, x_n\}$, $\xi_i \in [x_i, x_{i+1}]$. Pentru $f : [a, b] \rightarrow \mathbb{R}$ notăm

$$\sigma(\delta, f) = \sum_{i=0}^{n-1} f(\xi_i)(x_{i+1} - x_i)$$

și o numim **suma Riemann** asociată funcției f , diviziunii δ și sistemului de puncte intermediare $\xi = \{\xi_i\}$. Se poate folosi și notația $\sigma(\delta, \xi, f)$.

Teorema 2 (Criteriul cu sume Riemann). Funcția $f : [a, b] \rightarrow \mathbb{R}$ este integrabilă Riemann pe intervalul $[a, b]$ dacă și numai dacă există un număr real α și pentru orice $\varepsilon > 0$ există $\eta > 0$ astfel încât pentru orice sumă Riemann $\sigma(\delta, f)$ în care $\|\delta\| < \eta$ avem $|\sigma(\delta, f) - \alpha| < \varepsilon$. Atunci $\int_a^b f(x) dx = \alpha$.

Corolar 1. Funcția $f : [a, b] \rightarrow \mathbb{R}$ este integrabilă Riemann dacă și numai dacă există numărul real α și pentru orice sir de sume Riemann $(\sigma(\delta_n, f))_{n \in \mathbb{N}}$ în care $\lim_{n \rightarrow \infty} \|\delta_n\| = 0$ avem $\lim_{n \rightarrow \infty} \sigma(\delta_n, f) = \alpha$. Atunci $\int_a^b f(x) dx = \alpha$.

Fie I un interval în \mathbb{R} . Vom nota

$$\mu(I) = \begin{cases} 0 & \text{dacă } I = \emptyset \\ b - a & \text{dacă } (a, b) \subset I \subset [a, b] \\ \infty & \text{dacă } I \text{ este nemărginit} \end{cases}$$

Definiție. Mulțimea $A \subset \mathbb{R}$ se numește **neglijabilă Lebesgue** dacă pentru orice $\varepsilon > 0$ există un sir $(I_n)_{n \in \mathbb{N}}$ de intervale astfel încât $A \subset \bigcup_{n=0}^{\infty} I_n$,

$$\sum_{n=0}^{\infty} \mu(I_n) < \varepsilon.$$

Observație. În definiția precedentă se poate presupune că toate intervalele sunt închise (deschise).

Este evident că orice mulțime cel mult numărabilă este neglijabilă și că orice submulțime a unei mulțimi neglijabile este neglijabilă. Există mulțimi neglijabile și nenumărabile.

Propoziția 2. Orice reuniune numărabilă de mulțimi neglijabile este o mulțime neglijabilă.

Teorema 3 (Criterionul lui Lebesgue). Funcția $f : [a, b] \rightarrow \mathbb{R}$ este integrabilă Riemann dacă și numai dacă este mărginită și mulțimea punctelor de discontinuitate este neglijabilă.

IX.3. Proprietăți ale funcțiilor integrabile și ale integralei

Propoziția 1. Dacă $f \in \mathcal{R}[a, b]$, atunci $|f| \in \mathcal{R}[a, b]$ și

$$\left| \int_a^b f(x) dx \right| \leq \int_a^b |f(x)| dx$$

Propoziția 2. Dacă $f, g \in \mathcal{R}[a, b]$ și $\alpha, \beta \in \mathbb{R}$, atunci $fg \in \mathcal{R}[a, b]$, $\alpha f + \beta g \in \mathcal{R}[a, b]$ și

$$\int_a^b (\alpha f(x) + \beta g(x)) dx = \alpha \int_a^b f(x) dx + \beta \int_a^b g(x) dx.$$

Propoziția 3. Dacă $f : [a, b] \rightarrow [c, d]$ este integrabilă iar $\phi : [c, d] \rightarrow \mathbb{R}$ este continuă, atunci $\phi \circ f$ este integrabilă.

Propoziția 4. Dacă $f \in \mathcal{R}[a, b]$ și $[c, d] \subset [a, b]$, atunci $f|_{[c, d]} \in \mathcal{R}[c, d]$.

Propoziția 5. Dacă $f : [a, b] \rightarrow \mathbb{R}$ și există $c \in (a, b)$ astfel încât $f|_{[a,c]}$ și $f|_{[c,b]}$ sunt integrabile, atunci funcția f este integrabilă și

$$\int_a^b f(x) dx = \int_a^c f(x) dx + \int_c^b f(x) dx.$$

Se spune că egalitatea precedentă exprimă aditivitatea integralei ca funcție de interval.

Vom spune că o proprietate punctuală este adevărată aproape peste tot (a.p.t.) dacă este adevărată cu excepția unei multimi neglijabile.

Propoziția 6. Fie $f, g \in \mathcal{R}[a, b]$ astfel încât $f(x) \leq g(x)$ a.p.t.. Atunci

$$\int_a^b f(x) dx \leq \int_a^b g(x) dx.$$

Propoziția 7. Fie $f, g \in \mathcal{R}[a, b]$ o funcție pozitivă pentru care există $x_0 \in [a, b]$ un punct în care f este continuă și $f(x_0) > 0$. Atunci $\int_a^b f(x) dx > 0$.

Teorema 1. Fie $f, g \in \mathcal{R}[a, b]$ astfel încât $g(x) \geq 0$ pentru orice $x \in [a, b]$ (respectiv $g(x) \leq 0$ pentru orice $x \in [a, b]$). Atunci există $\lambda \in \left[\inf_{x \in [a,b]} f(x), \sup_{x \in [a,b]} f(x) \right]$ astfel încât

$$\int_a^b f(x) g(x) dx = \lambda \int_a^b g(x) dx.$$

Corolar. Fie $g \in \mathcal{R}[a, b]$ o funcție pozitivă (negativă) și $f : [a, b] \rightarrow \mathbb{R}$ o funcție continuă. Atunci există $\xi \in [a, b]$ astfel încât

$$\int_a^b f(x) g(x) dx = f(\xi) \int_a^b g(x) dx.$$

Propoziția 8. Dacă $f \in \mathcal{R}[a, b]$, atunci funcția $F : [a, b] \rightarrow \mathbb{R}$,

$$F(x) = \int_a^x f(t) dt$$

este continuă.

Teorema 2 (Formula lui Bonnet). Fie $f, g \in \mathcal{R}[a, b]$, g monotonă. Atunci există $\xi \in [a, b]$ astfel încât

$$\int_a^b f(x)g(x)dx = g(a)\int_a^\xi f(x)dx + g(b)\int_\xi^b f(x)dx$$

IX.4. Primitive

Definiție. Se spune că funcția $f : [a, b] \rightarrow \mathbb{R}$ **admit primitive** dacă există $F : [a, b] \rightarrow \mathbb{R}$ o funcție derivabilă astfel încât $F'(x) = f(x)$ pentru orice $x \in [a, b]$.

Propoziție. Dacă $f : [a, b] \rightarrow \mathbb{R}$ admite primitive și F_1, F_2 sunt două primitive atunci $F_1 - F_2$ este o constantă.

Teorema 1. Dacă $f \in \mathcal{R}[a, b]$ și $F : [a, b] \rightarrow \mathbb{R}$, $F(x) = \int_a^x f(t)dt$,

atunci F este derivabilă în orice punct x_0 în care f este continuă și $F'(x_0) = f(x_0)$.

Corolar. Orice funcție continuă admite primitive.

Teorema 2 (Formula Leibniz-Newton). Dacă $f \in \mathcal{R}[a, b]$ și admite primitive, atunci $\int_a^b f(x)dx = F(b) - F(a)$ pentru orice primitivă F .

Corolar. Fie $f, g : [a, b] \rightarrow \mathbb{R}$ funcții derivabile astfel încât $f', g' \in \mathcal{R}[a, b]$. Atunci

$$\int_a^b f(x)g'(x)dx = f(b)g(b) - f(a)g(a) - \int_a^b f'(x)g(x)dx.$$

IX.5. Siruri de funcții integrabile

Teorema 1. Fie $(f_n)_{n \in \mathbb{N}}$ un sir de funcții integrabile pe intervalul $[a, b]$, sir uniform convergent către funcția f . Atunci $f \in \mathcal{R}[a, b]$ și

$$\lim_{n \rightarrow \infty} \int_a^b f_n(x)dx = \int_a^b f(x)dx.$$

Teorema 2. Fie $(f_n)_{n \in \mathbb{N}}$ un şir monoton de funcţii integrabile pe intervalul $[a, b]$, şir punctual convergent către funcţia f . Atunci

$$\lim_{n \rightarrow \infty} \int_a^b f_n(x) dx = \int_a^b f(x) dx.$$

IX.6. Formule de schimbare de variabilă

Teorema 1 (Formula de schimbare de variabilă). Fie $f : [a, b] \rightarrow \mathbb{R}$ o funcţie continuă şi $\phi : [\alpha, \beta] \rightarrow [a, b]$ o funcţie derivabilă cu ϕ' integrabilă şi $\phi(\alpha) = a, \phi(\beta) = b$. Atunci

$$\int_a^b f(x) dx = \int_\alpha^\beta f(\phi(t)) \cdot \phi'(t) dt.$$

Teorema 2 (Formula de schimbare de variabilă). Fie $f \in \mathcal{R}[a, b]$ şi $\phi : [\alpha, \beta] \rightarrow [a, b]$ o funcţie strict monotonă, derivabilă, cu derivata continuă şi $\phi([\alpha, \beta]) = [a, b]$. Atunci

$$\int_\alpha^\beta f(\phi(t)) \cdot \phi'(t) dt = \int_{\phi(\alpha)}^{\phi(\beta)} f(x) dx.$$

X. INTEGRALA RIEMANN IMPROPRIE

Fie $A \in \{I \mid I \text{ interval necompact}\} \cup \left\{ I \setminus \{c\} \mid I \text{ interval, } c \in \overset{\circ}{I} \right\}$, şi

$f : A \rightarrow \mathbb{R}$ o funcţie integrabilă Riemann pe orice interval compact inclus în A . Prezentăm o anumită accepţiune a integralei funcţiei f pe mulţimea A , cazurile definindu-se prin intermediul celui în care $A \in \{[a, b] \mid b \in \bar{\mathbb{R}}\} \cup \{(a, b] \mid a \in \bar{\mathbb{R}}\}$.

Fie $a \in \mathbb{R}$, $b \in \bar{\mathbb{R}}$ şi $f : [a, b] \rightarrow \mathbb{R}$ o funcţie integrabilă Riemann pe orice interval $[a, c]$, $c \in [a, b)$ şi $F(x) = \int_a^x f(t) dt$, $x \in [a, b)$.

Definiție. Perechea (f, F) se numește **integrală Riemann impropriă** a lui f și se notează

$$\int_a^b f(x)dx$$

Se spune că integrală impropriă, $\int_a^b f(x)dx$, este **convergentă** dacă există

$$\lim_{\substack{x \rightarrow b \\ x < b}} \int_a^x f(t)dt \in \mathbb{R}. \text{ Această limită se notează tot cu } \int_a^b f(x)dx. \text{ Dacă limita}$$

precedentă nu există în \mathbb{R} , se spune că integrală considerată este **divergentă**.

Dacă $b \in \mathbb{R}$, integrală impropriă precedentă se notează uneori $\int_a^{b-0} f(x)dx$.

O definiție analogă se dă pentru integrală pe intervale de forma $(a, b]$, $a \in \bar{\mathbb{R}}$.

Exemple.

1. Fie $q > 0$, $a \in \mathbb{R}$ și $\int_a^\infty q^x dx$. Din

$$\int_a^x q^t dt = \begin{cases} x - a & \text{pentru } q = 1 \\ \frac{1}{\ln q} (q^x - q^a) & \text{pentru } q \neq 1 \end{cases}$$

rezultă că integrală considerată este convergentă dacă $q \in (0, 1)$ și divergentă dacă $q \in [1, \infty)$.

2. Fie $\lambda > 0$, $b > 0$ și $\int_0^b \frac{dx}{x^\lambda}$. Din

$$\int_x^b \frac{dt}{t^\lambda} = \begin{cases} \ln b - \ln x & \text{pentru } \lambda = 1 \\ \frac{1}{1-\lambda} (b^{1-\lambda} - x^{1-\lambda}) & \text{pentru } \lambda \neq 1 \end{cases}$$

rezultă că integrală considerată este convergentă dacă $\lambda \in (0, 1)$ și divergentă dacă $\lambda \in [1, \infty)$.

Observații.

1. Integrala $\int\limits_a^{b-0} f(x)dx$ are aceeași natură (este convergentă sau divergentă) cu integrala $\int\limits_c^{b-0} f(x)dx$, unde $c \in [a, b]$.
2. Dacă $b \in \mathbb{R}$ iar funcția $f : [a, b] \rightarrow \mathbb{R}$ este integrabilă pe orice interval compact inclus în $[a, b]$ și este mărginită, atunci prelungirea ei în b cu o valoare arbitrară este o funcție mărginită, continuă aproape peste tot, deci integrabilă pe $[a, b]$. Astfel, integrala impropriă $\int\limits_a^{b-0} f(x)dx$ este convergentă și valoarea ei este integrala pe $[a, b]$ a unei prelungiri a funcției f . Se poate spune că integrala este proprie. Pentru $b \in \mathbb{Z}$ vom avea deci de studiat doar cazul în care funcția de integrat este nemărginită.

Definiție. Se spune că integrala impropriă $\int\limits_a^b f(x)dx$ este **absolut convergentă** dacă $\int\limits_a^b |f(x)|dx$ este convergentă.

Teorema 1. Dacă integrala impropriă $\int\limits_a^b f(x)dx$ este absolut convergentă,

atunci ea este convergentă și

$$\left| \int\limits_a^b f(x)dx \right| \leq \int\limits_a^b |f(x)|dx .$$

Teorema 2. Fie $f, g : [a, b] \rightarrow \mathbb{R}_+$, funcții integrabile pe orice interval compact inclus în $[a, b]$.

- i) Dacă $f(x) \leq g(x)$ pentru orice $x \in [a, b]$ și dacă integrala $\int\limits_a^b g(x)dx$ este convergentă, atunci $\int\limits_a^b f(x)dx$ este convergentă și $\int\limits_a^b f(x)dx \leq \int\limits_a^b g(x)dx$.

ii) Dacă $g(x) > 0$ orice $x \in [a, b)$ și există

$$\lim_{\substack{x \rightarrow b \\ x < b}} \frac{f(x)}{g(x)} = l,$$

atunci

a) dacă $l \in (0, \infty]$ și $\int_a^b g(x)dx$ este divergentă, atunci $\int_a^b f(x)dx$ este

divergentă;

b) dacă $l \in [0, \infty)$ și $\int_a^b g(x)dx$ este convergentă, atunci $\int_a^b f(x)dx$ este

convergentă;

Exemplu. Fie $n \geq 1$ și $\Gamma(n) = \int_0^\infty e^{-x} x^{n-1} dx$. Deoarece

$(x^2 + 1)e^{-x} x^{n-1} = 0$, iar $\int_0^\infty \frac{1}{1+x^2} dx$ este convergentă, rezultă că integrala considerată este convergentă.

Teorema 3 (criteriul intergal pentru serii). Fie $f : [1, \infty) \rightarrow \mathbb{R}_+$ o funcție descrescătoare. Atunci $\int_1^\infty f(x)dx$ este convergentă dacă și numai dacă seria $\sum_{n \geq 1} f(n)$ este convergentă.

Teorema 4 (criteriul lui Abel). Fie $f, g : [a, b] \rightarrow \mathbb{R}$, f integrabilă pe orice interval compact din $[a, b]$, g monotonă și mărginită. Dacă $\int_a^b f(x)dx$ este convergentă, atunci $\int_a^b f(x)g(x)dx$ este de asemenea convergentă.

Teorema 5 (criteriul lui Dirichlet). Fie $f, g : [a, b] \rightarrow \mathbb{R}$, f integrabilă pe orice interval compact din $[a, b]$ astfel încât funcția F , $F(x) = \int_a^x f(t)dt$ este mărginită iar g este monotonă și $\lim_{\substack{x \rightarrow b \\ x < b}} g(x) = 0$. Atunci $\int_a^b f(x)g(x)dx$ este convergentă.

Exemplu. Integrala $\int_0^\infty \frac{\sin x}{x} dx$, numită integrala lui Dirichlet, este convergentă și are valoarea $\frac{\pi}{2}$.

Fie $a, b \in \mathbb{R}$, $c \in (a, b)$ și $f : [a, b] \setminus \{c\} \rightarrow \mathbb{R}$ o funcție integrabilă Riemann pe orice interval compact inclus în $[a, b] \setminus \{c\}$.

Definiție. Dacă $\int_a^c f(x)dx$ și $\int_c^b f(x)dx$ sunt convergente, atunci integrala improprie $\int_a^b f(x)dx$ se numește **convergentă** și

$$\int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx.$$

În caz contrar integrala se numește **divergentă**.

Exemplu. Integrala $\int_{-1}^1 \frac{1}{\sqrt{|x|}} dx$ este convergentă iar $\int_{-1}^1 \frac{1}{x} dx$ este divergentă.

Definiție. Dacă integrala din definiția precedentă este divergentă și există

$$\lim_{\varepsilon \rightarrow 0} \left(\int_a^{c-\varepsilon} f(x)dx + \int_{c+\varepsilon}^b f(x)dx \right) \in \mathbb{R}$$

se spune că integrala este **convergentă în sensul valorii principale** și se notează

$$v.p. \int_a^b f(x)dx = \lim_{\varepsilon \rightarrow 0} \left(\int_a^{c-\varepsilon} f(x)dx + \int_{c+\varepsilon}^b f(x)dx \right).$$

Exemplu. Integrala $\int_{-1}^1 \frac{1}{x} dx$ este divergentă dar $v.p. \int_{-1}^1 \frac{1}{x} dx = 0$.

Definiție. Fie $a, b \in \overline{\mathbb{R}}$ și $f : (a, b) \rightarrow \mathbb{R}$ integrabilă pe orice interval compact inclus în (a, b) . Integrala improprie $\int_a^b f(x)dx$ se numește **convergentă**

dacă există $c \in (a, b)$ astfel încât integralele $\int_a^c f(x)dx$ și $\int_c^b f(x)dx$ sunt

convergente și $\int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx$.

În caz contrar, integrala se numește **divergentă**.

Exemple.

1. Integrala $\int_{-\infty}^{\infty} e^{-x^2} dx$, numită integrala Euler-Poisson, este convergentă căci $e^{-x^2} \leq \frac{1}{1+x^2}$ pentru orice x . Valoarea acestei integrale este $\sqrt{\pi}$.
2. Integrala $\Gamma(t) = \int_0^{\infty} e^{-x} x^{t-1} dx$ este convergentă pentru $t > 0$ și divergentă pentru $t \leq 0$.

Definiție. Dacă $\int_{-\infty}^{\infty} f(x)dx$ este divergentă și există $\lim_{r \rightarrow \infty} \int_{-r}^r f(x)dx \in \mathbb{R}$,

atunci integrala considerată se numește **convergentă în sensul valorii principale** și

$$v.p. \int_{-\infty}^{\infty} f(x)dx = \lim_{r \rightarrow \infty} \int_{-r}^r f(x)dx.$$

Exemplu. Integrala $\int_{-\infty}^{\infty} \frac{x}{1+x^2} dx$ este divergentă și $v.p. \int_{-\infty}^{\infty} \frac{x}{1+x^2} dx = 0$.

XI. SPAȚIUL

XI.1. Structura algebrică și structura topologică

Fie $m \in \mathbb{N}$, $m > 1$. Se notează

$$\mathbb{R}^m = \{(x_1, \dots, x_m) | x_i \in \mathbb{R}\}$$

și se spune că \mathbb{R}^m este mulțimea tuturor sistemelor ordonate de m numere reale. Atunci \mathbb{R}^m este de fapt produsul cartezian $\mathbb{R} \times \mathbb{R} \times \dots \times \mathbb{R}$ (de m ori). Elementele lui \mathbb{R}^m se numesc vectori. Doi vectori (x_1, \dots, x_m) , (y_1, \dots, y_m) se numesc egali dacă $x_i = y_i$ pentru orice $i \in \{1, \dots, m\}$ și se scrie $(x_1, \dots, x_m) = (y_1, \dots, y_m)$.

Structuri algebrice

Fie $x = (x_1, \dots, x_m)$, $y = (y_1, \dots, y_m)$ și $\lambda \in \mathbb{Z}$. Se notează

$$x + y = (x_1 + y_1, \dots, x_m + y_m),$$

$$\lambda x = (\lambda x_1, \dots, \lambda x_m)$$

și se numesc suma vectorilor x și y , respectiv produsul dintre numărul λ și vectorul x .

Împreună cu operațiile precedente \mathbb{R}^m este spațiu vectorial (liniar) peste corpul \mathbb{R} . Elementul neutru față de ordonare este $0 = (0, \dots, 0)$ iar $-x = (-x_1, \dots, -x_m)$. Se notează $e_i = (0, \dots, 0, 1, 0, \dots, 0)$ unde 1 se află pe locul i, $i \in \{1, \dots, m\}$. Multimea $\{e_1, \dots, e_m\}$ este o bază algebrică în \mathbb{R}^m .

Structura topologică

Definiție. Aplicația $p : \mathbb{R}^m \rightarrow \mathbb{R}$ se numește **normă** (pe \mathbb{R}^m) dacă

- i) $p(x) = 0 \Rightarrow x = 0$;
- ii) $p(x + y) \leq p(x) + p(y)$, $\forall x, y \in \mathbb{R}^m$;
- iii) $p(\lambda x) = |\lambda| p(x)$, $\forall x \in \mathbb{R}^m$, $\forall \lambda \in \mathbb{R}$.

Se notează de obicei $p(x) = \|x\|$ și se citește „normă lui x” iar proprietățile precedente se scriu:

$$\begin{aligned}\|x\| &= 0 \Rightarrow x = 0; \\ \|x + y\| &\leq \|x\| + \|y\|, \quad \forall x, y \in \mathbb{R}^m; \\ \|\lambda x\| &= |\lambda| \|x\|, \quad \forall x \in \mathbb{R}^m, \quad \forall \lambda \in \mathbb{R}.\end{aligned}$$

Observații

$$\begin{aligned}\|x\| &\geq 0, \quad \forall x \in \mathbb{R}^m; \\ \|x\| &= 0 \Leftrightarrow x = 0; \\ \|x\| - \|y\| &\leq \|x - y\|.\end{aligned}$$

Definiție. Normele $p, q : \mathbb{R}^m \rightarrow \mathbb{R}$ se numesc echivalente dacă există $\alpha, \beta > 0$ astfel încât

$$\alpha p(x) \leq q(x) \leq \beta p(x), \quad \forall x \in \mathbb{R}^m.$$

În spațiul \mathbb{R}^m orice două norme sunt echivalente. O normă va juca rolul pe care îl are în \mathbb{R} funcția modul.

Dacă $x \mapsto \|x\|$ este o normă pe \mathbb{R}^m , atunci aplicația $d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_+$, $d(x, y) = \|x - y\|$ este o distanță pe \mathbb{R}^m numită **distanța generată de normă** $\|\cdot\|$.

Aplicația d are deci proprietățile:

$$\begin{aligned}d(x, y) &= d(y, x), \\ d(x, y) &= 0 \Leftrightarrow x = y, \\ d(x, z) &\leq d(x, y) + d(y, z).\end{aligned}$$

Fie $\|\cdot\|$ o normă pe \mathbb{R}^m , $r > 0$ și $x_0 \in \mathbb{R}^m$. Se notează

$$B(x_0, r) = \left\{ x \in \mathbb{R}^m \mid \|x - x_0\| < r \right\}$$

$$\overline{B}(x_0, r) = \left\{ x \in \mathbb{R}^m \mid \|x - x_0\| \leq r \right\}$$

și se numesc **bila deschisă**, respectiv **bila închisă**, de centru x_0 și rază r .

Definiție. Mulțimea $V \subset \mathbb{R}^m$ se numește **vecinătate** pentru punctul x_0 dacă există $r > 0$ astfel încât $B(x_0, r) \subset V$.

Observații:

1. Deoarece pe \mathbb{R}^m orice două norme sunt echivalente, definiția precedentă nu depinde de normă considerată, deci, dacă V este o vecinătate a lui x_0 în raport cu o normă, atunci ea este vecinătate în raport cu orice altă normă.
2. Orice bilă deschisă sau închisă este vecinătate pentru centrul x_0 . Orice bilă deschisă este vecinătate pentru fiecare punct al său. Bila închisă $\overline{B}(x_0, r)$ nu este vecinătate pentru punctele z , $\|z - x_0\| = r$.

Familia V_{x_0} a tuturor vecinătăților punctului x_0 are proprietățile:

- i) $V \in V_{x_0} \Rightarrow x_0 \in V$,
- ii) $V \in V_{x_0}$ și $V \subset W \Rightarrow W \in V_{x_0}$,
- iii) $V_1, V_2 \in V_{x_0} \Rightarrow V_1 \cap V_2 \in V_{x_0}$,
- iv) $\forall V \in V_{x_0}, \exists W \in V_{x_0}, \forall y \in W \Rightarrow V \in V_y$.

Definiție. Mulțimea $D \subset \mathbb{R}^m$ se numește **deschisă** dacă este vecinătate pentru orice punct al său.

Observații:

1. O mulțime este deschisă dacă și numai dacă este o reuniune de bile deschise.
2. Familia τ a mulțimilor deschise are proprietățile:
 - $\mathbb{R}^m, \emptyset \in \tau$,
 - $G_1, G_2 \in \tau \Rightarrow G_1 \cap G_2 \in \tau$,
 - $(G_i)_{i \in I} \subset \tau \Rightarrow \bigcup_{i \in I} G_i \in \tau$.

deci constituie o topologie pe \mathbb{R}^m .

Definiție. Mulțimea $F \subset \mathbb{R}^m$ se numește **închisă** dacă $\mathbb{R}^m \setminus F$ este deschisă.

Exemplu. Orice punct este o mulțime închisă. Orice mulțime finită este închisă. Orice bilă închisă este o mulțime închisă.

Observație. Familia mulțimilor închise are proprietățile:

- i) \mathbb{R}^m și \emptyset sunt mulțimi închise.
- ii) Dacă F_1 și F_2 sunt mulțimi închise, atunci $F_1 \cup F_2$ este mulțime închisă.
- iii) Dacă $(F_i)_{i \in I}$ este o familie de mulțimi închise, atunci $\bigcap_{i \in I} F_i$ este o mulțime închisă.

Structura topologică a unei mulțimi

Definiție. Punctul x se numește **interior** mulțimii $A \subset \mathbb{R}^m$ dacă A este o vecinătate a lui x .

Mulțimea punctelor interioare mulțimii A se numește interiorul lui A și se notează $\overset{\circ}{A}$ sau $\text{int } A$.

O mulțime este deschisă dacă și numai dacă coincide cu interiorul ei.

Definiție. Se spune că punctul x este **aderent** mulțimii A dacă pentru orice vecinătate V a lui x avem $V \cap A \neq \emptyset$.

Mulțimea punctelor aderente lui A se numește **aderență (închiderea)** lui A și se notează \overline{A} .

Observații:

$$\begin{aligned} A &\subset \overline{A}, \\ {}^0 CA &= \overline{CA}, \\ {}^0 C\overline{A} &= \overline{CA}. \end{aligned}$$

O mulțime este închisă dacă și numai dacă coincide cu aderența sa.

Definiție. Sirul $(x_n)_{n \in \mathbb{N}}$ din \mathbb{R}^m se numește **convergent** dacă există $x \in \mathbb{R}^m$ și pentru orice $\varepsilon > 0$ există $n_\varepsilon \in \mathbb{N}$ astfel încât, pentru orice $n \geq n_\varepsilon$ avem $\|x_n - x\| < \varepsilon$. Elementul x se numește **limita** sirului, $x = \lim_{n \rightarrow \infty} x_n$.

Observații

1. Sirul $(x_n)_{n \geq 1}$ este convergent dacă și numai dacă există $x \in \mathbb{R}^m$ astfel încât $\lim_{n \rightarrow \infty} \|x_n - x\| = 0$.
2. Definiția nu depinde de norma considerată pe \mathbb{R}^m .
3. Limita unui sir convergent este unică.
4. Dacă $x_n = (x_1^{(n)}, \dots, x_m^{(n)})$ și $x = (x_1, \dots, x_m)$, atunci $\lim_{n \rightarrow \infty} x_n = x$ dacă și numai dacă $\lim_{n \rightarrow \infty} x_i^{(n)} = x_i$ pentru orice $i \in \{1, \dots, m\}$.

5. Orice sir convergent este marginit adica există $M > 0$ și pentru orice $n \in \mathbb{N}$, $\|x_n\| \leq M$.
6. Pentru orice sir marginit există un subșir convergent.
7. Dacă $\lim_{n \rightarrow \infty} x_n = x$, $\lim_{n \rightarrow \infty} y_n = y$, atunci $\lim_{n \rightarrow \infty} (\alpha x_n + \beta y_n) = \alpha x + \beta y$.
8. Dacă $\lim_{n \rightarrow \infty} x_n = x$, atunci $\lim_{n \rightarrow \infty} \|x_n\| = \|x\|$.
9. Punctul x este aderent mulțimii $A \subset \mathbb{R}^m$ dacă și numai dacă există un sir de elemente din A convergent către x .

Definiție. Se spune că $x \in \mathbb{R}^m$ este **punct de acumulare** pentru mulțimea $A \subset \mathbb{R}^m$ dacă pentru orice vecinătate V a lui x avem $A \cap V \setminus \{x\} \neq \emptyset$.

Observații

1. Elementul x este punct de acumulare pentru mulțimea A dacă și numai dacă există un sir $(a_n)_{n \in \mathbb{N}}$ de elemente din $A \setminus \{x\}$ astfel încât $\lim_{n \rightarrow \infty} a_n = x$.
2. Mulțimea A este închisă dacă și numai dacă își conține punctele de acumulare.
3. O mulțime finită nu are puncte de acumulare.
4. Dacă $(a_n)_{n \in \mathbb{N}}$ este un sir convergent, $\lim_{n \rightarrow \infty} a_n = a$ și $a_n \neq a$, $\forall n \in \mathbb{N}$, atunci a este punct de acumulare pentru mulțimea $\{a_n | n \in \mathbb{N}\}$.
5. Orice mulțime infinită și marginită are cel puțin un punct de acumulare.

Definiție. Punctul x se numește **punct frontieră** pentru mulțimea $A \subset \mathbb{R}^m$ dacă este aderent atât mulțimii A cât și complementarei ei.

Mulțimea punctelor frontieră se notează ∂A (sau $\text{fr } A$) și se numește **frontiera** lui A , $\partial A = \overline{A} \cap \overline{C_A}$.

Exemplu: $\partial B(x_0, r) = \{x | \|x - x_0\| = r\}$.

Definiție. Mulțimea $A \subset \mathbb{R}^m$ se numește **marginită** dacă există $M > 0$ astfel încât $\|a\| \leq M$, $\forall a \in A$.

Definiție. Mulțimea $M \subset \mathbb{R}^m$ se numește **compactă** dacă și numai dacă este marginită și închisă.

O mulțime A este compactă dacă și numai dacă pentru orice sir $(a_n)_{n \in \mathbb{N}}$ de elemente din A , există un subșir $(a_{n_k})_{k \in \mathbb{N}}$ și există $a \in A$ astfel încât $\lim_{k \rightarrow \infty} a_{n_k} = a$.

XI.2. Integrala Riemann cu parametru

Fie $I \subset \mathbb{R}$ un interval și $F : [a, b] \times I \rightarrow \mathbb{R}$ astfel încât pentru orice $t \in I$ funcția $x \mapsto F(x, t)$ este integrabilă Riemann pe $[a, b]$. Fie $u, v : I \rightarrow [a, b]$ și

$$f(t) = \int_{u(t)}^{v(t)} F(x, t) dx \quad (1)$$

Funcția f se numește **integrală cu parametru**.

Teorema 1. Dacă funcția F este continuă pe $[a, b] \times I$ și u, v sunt continue pe I , atunci f este continuă pe I .

Teorema 2. Fie $F : [a, b] \times I \rightarrow \mathbb{R}$ o funcție continuă, derivabilă parțial în raport cu t și astfel încât $\frac{\partial F}{\partial t} : [a, b] \times I \rightarrow \mathbb{R}$ să fie continuă. Atunci $f(t) = \int_a^b F(x, t) dx$ este derivabilă, cu derivata continuă și $f'(t) = \int_a^b \frac{\partial F}{\partial t}(x, t) dx$ (formula lui Leibniz).

Teorema 3. Fie $F : [a, b] \times I \rightarrow \mathbb{R}$ o funcție continuă, derivabilă parțial în raport cu t și astfel încât $\frac{\partial F}{\partial t} : [a, b] \times I \rightarrow \mathbb{R}$ să fie continuă. Fie $u, v : I \rightarrow [a, b]$ funcții derivabile și $f(t) = \int_{u(t)}^{v(t)} F(x, t) dx$. Atunci f este derivabilă și

$$f'(t) = \int_{u(t)}^{v(t)} \frac{\partial F}{\partial t}(x, t) dx + F(v(t), t)v'(t) - F(u(t), t)u'(t).$$

Fie $b \in \bar{\mathbb{R}}$ și $F : [a, b] \times I \rightarrow \mathbb{R}$ astfel încât pentru orice $t \in \mathbb{R}$ funcția $x \mapsto F(x, t)$ este integrabilă pe orice interval compact inclus în $[a, b]$.

Definiție. Se spune că integrala $\int_a^b F(x, t) dx$ este **convergentă** dacă pentru orice $t \in I$ există $\lim_{y \rightarrow b} \int_a^y F(x, t) dx = f(t) \in \mathbb{R}$.

Se spune că integrala $\int_a^b F(x, t) dx$ este **uniform convergentă** dacă este convergentă și pentru orice $\varepsilon > 0$ există $c \in [a, b]$ astfel încât $\forall y \in (c, b)$

$$\left| \int_a^y F(x, t) dx - f(t) \right| < \varepsilon, \quad \forall t \in I.$$

Propoziția 1. Dacă există $g : [a, b] \rightarrow \mathbb{R}_+$ astfel încât integrala $\int_a^b g(x)dx$ este convergentă și $|F(x, t)| \leq g(x)$, $\forall x \in [a, b]$, $\forall t \in I$, atunci integrala $\int_a^b F(x, t)dx$ este uniform convergentă.

Propoziția 2. Dacă F este continuă și dacă integrala $\int_a^b F(x, t)dx$ este uniform convergentă, atunci funcția f este continuă.

Propoziția 3. Dacă I este mărginit, F este continuă, $\frac{\partial F}{\partial t}$ există și este continuă, $\int_a^b F(x, t)dx$ este convergentă iar $\int_a^b \frac{\partial F}{\partial t}(x, t)dx$ este uniform convergentă, atunci f este derivabilă și $f'(t) = \int_a^b \frac{\partial F}{\partial t}(x, t)dx$.

XI.3. Drumuri, curbe, integrala curbilinie

Fie $A \subset \mathbb{R}^m$ o mulțime deschisă.

Definiție. Se numește **drum** în A orice funcție continuă $\gamma : [a, b] \subset \mathbb{R} \rightarrow A$. Punctul $\gamma(a)$ se numește originea drumului iar $\gamma(b)$ capătul lui. Drumul se numește **închis** dacă $\gamma(a) = \gamma(b)$.

Mulțimea $\gamma[a, b] = \{\gamma(t) | t \in [a, b]\}$ se numește **suportul sau imaginea** drumului γ .

Drumul γ se numește **simplu** dacă funcțiile $\gamma|_{(a, b]}$ și $\gamma|_{[a, b)}$ sunt injective.

Definiție. Se spune că drumul γ este **de clasă C^1** dacă toate funcțiile γ_i sunt derivabile cu derivata continuă.

Dacă există o diviziune $a = t_0 < t_1 < \dots < t_k = b$ astfel încât $\gamma_i|_{[t_j, t_{j+1}]}$ este de clasă C^1 pentru orice $i \in \{1, \dots, n\}$ și orice $j \in \{0, \dots, k-1\}$ se spune că drumul γ este **de clasă C^1 pe porțiuni**.

Exemplu. Drumul $\gamma : [0, 1] \rightarrow \mathbb{R}^n$ se numește **poligonal** dacă există $\delta = \{t_0, \dots, t_k\}$ o diviziune a intervalului $[0, 1]$ astfel încât

$$\gamma(t) = \frac{t_{i+1} - t}{t_{i+1} - t_i} \gamma(t_i) + \frac{t - t_i}{t_{i+1} - t_i} \gamma(t_{i+1}), \quad t \in [t_i, t_{i+1}],$$

pentru orice $i \in \{0, \dots, k-1\}$. Un astfel de drum, al cărui suport este o linie poligonală este de clasă C^1 pe porțiuni.

Fie γ_1, γ_2 drumuri în A , $\gamma_1 : [a, b] \rightarrow A$, $\gamma_2 : [c, d] \rightarrow A$. Dacă $\gamma_1(b) = \gamma_2(c)$ se notează $\gamma_1 \vee \gamma_2 : [a, b+d-c] \rightarrow A$,

$$\gamma_1 \vee \gamma_2(t) = \begin{cases} \gamma_1(t) & \text{dacă } t \in [a, b] \\ \gamma_2(c+b-t) & \text{dacă } t \in [b, b+d-c] \end{cases}$$

și se numește juxtapunerea drumurilor γ_1 și γ_2 .

Imaginea drumului $\gamma_1 \vee \gamma_2$ este reuniunea imaginilor drumurilor γ_1, γ_2 .

Definiție. Drumul $\gamma^{-1} : [a, b] \rightarrow A$, $\gamma^{-1}(t) = \gamma(a+b-t)$, se numește *inversul drumului γ* .

Imaginea lui γ^{-1} coincide cu imaginea lui γ .

Fie $P, Q : A \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ funcții continue și $\gamma = (\gamma_1, \gamma_2)$, $\gamma_i : [a, b] \rightarrow A$,

$i = 1, 2$ un drum de clasă C^1 pe porțiuni.

Definiție. Numărul

$$\int_a^b P(\gamma_1(t), \gamma_2(t)) \gamma'_1(t) dt + \int_a^b Q(\gamma_1(t), \gamma_2(t)) \gamma'_2(t) dt$$

se numește *integrala formei diferențiale* $\omega = Pdx + Qdy$ și se notează $\int_{\gamma} Pdx + Qdy$ (pe scurt $\int_{\gamma} \omega$).

Analog, dacă $P, Q, R : A \subset \mathbb{R}^3 \rightarrow \mathbb{R}$ sunt funcții continue și $\gamma = (\gamma_1, \gamma_2, \gamma_3)$, $\gamma_i : [a, b] \rightarrow A$, $i = 1, 2, 3$ este un drum de clasă C^1 pe porțiuni, atunci numărul

$$\int_a^b (P \circ \gamma)(t) \gamma'_1(t) dt + \int_a^b (Q \circ \gamma)(t) \gamma'_2(t) dt + \int_a^b (R \circ \gamma)(t) \gamma'_3(t) dt$$

se numește *integrala formei diferențiale* $\omega = Pdx + Qdy + Rdz$ și se notează $\int_{\gamma} Pdx + Qdy + Rdz$, pe scurt $\int_{\gamma} \omega$.

Noțiunea poate fi evident introdusă pentru drumurile în \mathbb{R}^m și forme diferențiale corespunzătoare. Au loc:

$$\int_{\gamma^{-1}} \omega = - \int_{\gamma} \omega,$$

$$\int_{\gamma_1 \vee \gamma_2} \omega = \int_{\gamma_1} \omega + \int_{\gamma_2} \omega.$$

Definiție. Forma diferențială $\omega = \sum_{i=1}^m \omega_i dx_i$ se numește **exactă** dacă există $f : A \rightarrow \mathbb{R}$ o funcție derivabilă cu toate derivatele parțiale continue astfel încât $\omega_i = \frac{\partial f}{\partial x_i}$, $\forall i \in \{1, \dots, m\}$. Scriem pe scurt $\omega = f'$.

Teorema 1 (independența de drum la integrala curbilinie). Fie $A \subset \mathbb{R}^m$ o mulțime deschisă și conexă și ω o formă diferențială pe A . Afirmațiile următoare sunt echivalente:

- i) ω este exactă.
- ii) $\int_{\gamma} \omega = 0$ pentru orice drum γ în A , închis și de clasă C^1 pe porțiuni.
- iii) $\int_{\lambda} \omega = \int_{\mu} \omega$ pentru orice drumuri λ, μ în A , de clasă C^1 pe porțiuni, având aceeași origine și același capăt.

Observații

1. Orice mulțime convexă este conexă. Amintim că o mulțime se numește convexă dacă odată cu orice două puncte din mulțime segmentul care unește aceste puncte este inclus în mulțime.
2. Pentru $m = 2$ forma diferențială $\omega Pdx + Qdy$ este exactă dacă există $f : A \rightarrow \mathbb{R}$ o funcție derivabilă cu derivatele parțiale continue astfel încât $P = \frac{\partial f}{\partial x}$, $Q = \frac{\partial f}{\partial y}$.

Definiție. Forma diferențială $\omega = \sum_{i=1}^m \omega_i dx_i$ se numește **închisă** dacă

$$\frac{\partial \omega_i}{\partial x_j}(x) = \frac{\partial \omega_j}{\partial x_i}(x) \text{ pentru orice } x \in A \text{ și orice } i, j \in \{1, \dots, m\}.$$

Observație. Orice formă diferențială ω , exactă și de clasă C^1 este închisă (rezultă din teorema Young).

Teorema 2 (Poincar). Dacă $A \subset \mathbb{R}^m$ este o mulțime deschisă și convexă, atunci orice formă diferențială închisă este exactă.

XI.4. Funcții derivabile

Fie $A \subset \mathbb{R}^m$ o mulțime deschisă.

Definiție. Se spune că funcția $f: A \rightarrow \mathbb{R}$ este **derivabilă Fréchet** (**diferențială**) în punctul $a \in A$ dacă există $(\lambda_1, \dots, \lambda_m) \in \mathbb{R}^m$ astfel încât

$$\lim_{x \rightarrow a} \frac{f(x) - f(a) - \sum_{i=1}^m \lambda_i (x_i - a_i)}{\|x - a\|} = 0.$$

Dacă există, vectorul $(\lambda_1, \dots, \lambda_m)$ este unic și se numește **derivata** funcției f în punctul a , $f'(a) = (\lambda_1, \dots, \lambda_m)$.

Definiția se scrie: $\forall \varepsilon > 0, \exists \eta > 0, \|x - a\| \leq \eta, x \in A \Rightarrow$

$$\left| f(x) - f(a) - \sum_{i=1}^m \lambda_i (x_i - a_i) \right| \leq \varepsilon \|x - a\|.$$

Teorema 1. Orice funcție derivabilă într-un punct este continuă în acel punct.

Propoziție. Fie $f, g: A \rightarrow \mathbb{R}$ derivabile în punctul $a \in A$ și $\alpha, \beta \in \mathbb{R}$. Atunci $\alpha f + \beta g$ este derivabilă în a și

$$(\alpha f + \beta g)'(a) = \alpha f'(a) + \beta g'(a).$$

Definiție. Se spune că funcția $f: A \subset \mathbb{R}^m \rightarrow \mathbb{R}$ este **derivabilă parțial** în raport cu variabila x_i în punctul $a \in A$ dacă există

$$\lim_{t \rightarrow a_i} \frac{f(a_1, \dots, a_{i-1}, t, a_{i+1}, \dots, a_m) - f(a_1, \dots, a_m)}{t - a_i} \in \mathbb{R}.$$

Această limită se notează cu $\frac{\partial f}{\partial x_i}(a)$ (sau $f'_{x_i}(a)$) și se numește **derivata parțială a funcției** f , **în raport cu variabila** x_i **în punctul** a .

În particular, dacă $m = 2$ și variabilele sunt notate x, y , atunci

$$\frac{\partial f}{\partial x}(a, b) = \lim_{x \rightarrow a} \frac{f(x, b) - f(a, b)}{x - a}$$

$$\frac{\partial f}{\partial y}(a, b) = \lim_{y \rightarrow b} \frac{f(a, y) - f(a, b)}{y - b}.$$

Observație. În acest caz particular dacă considerăm funcția de o variabilă $g : \{x | (x, b) \in A\} \rightarrow \mathbb{R}$, $g(x) = f(x, b)$, atunci f este derivabilă parțial în raport cu x în punctul (a, b) dacă și numai dacă g este derivabilă în a . Atunci $g'(a) = \frac{\partial f}{\partial x}(a, b)$.

Observație. Dacă $f : A \subset \mathbb{R}^m \rightarrow \mathbb{R}$ este derivabilă în punctul $a \in A$, atunci f este derivabilă parțial în raport cu fiecare variabilă în punctul a și

$$f'(a) = \left(\frac{\partial f}{\partial x_1}(a), \dots, \frac{\partial f}{\partial x_m}(a) \right).$$

Se notează uneori $\nabla f(a) = \left(\frac{\partial f}{\partial x_1}(a), \dots, \frac{\partial f}{\partial x_m}(a) \right)$ și se numește

gradientul lui f în punctul a .

Teorema 2 (Lagrange). Fie $f : A \subset \mathbb{R}^m \rightarrow \mathbb{R}$ o funcție derivabilă parțial în raport cu fiecare variabilă în orice punct din $B(x_0, r) \subset A$. Atunci pentru orice $a, b \in B(x_0, r)$, $a = (a_1, \dots, a_m)$, $b = (b_1, \dots, b_m)$ și orice $i \in \{1, \dots, m\}$ există $c_i \in [a_i, b_i]$ astfel încât

$$f(b) - f(a) = \sum_{i=1}^m \frac{\partial f}{\partial x_i}(y^{(i)})(b_i - a_i)$$

unde $y^{(i)} = (a_1, \dots, a_{i-1}, c_i, b_{i+1}, \dots, b_m)$.

Teorema 3. Fie $f : A \subset \mathbb{R}^m \rightarrow \mathbb{R}$ derivabilă parțial în raport cu fiecare variabilă pe o vecinătate a unui punct $a \in A$ și cu derivatele parțiale continue în a . Atunci f este derivabilă în a .

Definiție. Se spune că aplicația $f : A \subset \mathbb{R}^m \rightarrow \mathbb{R}$ este **derivabilă parțial în raport cu x_j** și apoi **în raport cu x_k în punctul $a \in A$** dacă există $B(a, r) \subset A$ astfel încât f este derivabilă parțial în raport cu x_j pe $B(a, r)$ iar $\frac{\partial f}{\partial x_j} : B(a, r) \rightarrow \mathbb{R}$ este derivabilă parțial în raport x_k cu în a .

Se notează

$$\frac{\partial^2 f}{\partial x_k \partial x_j}(a) = \frac{\partial}{\partial x_k} \left(\frac{\partial f}{\partial x_j} \right)(a).$$

Dacă $j = k$ se notează

$$\frac{\partial^2 f}{\partial x_k^2}(a) = \frac{\partial^2 f}{\partial x_k \partial x_k}(a).$$

Teorema 4 (Young). Dacă funcția $f : A \subset R^m \rightarrow R$ admite derivatele parțiale $\frac{\partial^2 f}{\partial x_k \partial x_j}$, $\frac{\partial^2 f}{\partial x_j \partial x_k}$ pe o vecinătate a punctului $a \in A$, derivate continue în a , atunci

$$\frac{\partial^2 f}{\partial x_k \partial x_j}(a) = \frac{\partial^2 f}{\partial x_j \partial x_k}(a).$$

Fie $A \subset \mathbb{R}$, $x_0 \in A$ un punct de acumulare pentru A și $f : A \rightarrow \mathbb{R}^m$, $f = (f_1, \dots, f_m)$, $f_i : A \rightarrow R$.

Definiție. Funcția f se numește **derivabilă** în punctul x_0 dacă $\forall i \in \{1, \dots, m\}$, f_i este derivabilă în x_0 . Atunci

$$f'(x_0) = (f'_1(x_0), f'_2(x_0), \dots, f'_m(x_0)).$$

Teorema 5. Fie $f : [a, b] \rightarrow \mathbb{R}^m$ continuă pe $[a, b]$, derivabilă pe (a, b) .

Atunci există $c \in (a, b)$ astfel încât

$$\|f(b) - f(a)\|_2 \leq \|f'(c)\|_2 (b - a).$$

Exemplu. Fie $f : [0, 1] \rightarrow \mathbb{R}^2$

$$f(x) = \begin{cases} (\sin 2\pi x, x(1-x)) & \text{dacă } x \neq 0 \\ (0, 0) & \text{dacă } x = 0 \end{cases}.$$

Atunci $f'(x) = (2\pi \cos 2\pi x, 1 - 2x)$.

Fie $A \subset \mathbb{R}^m$ o mulțime deschisă și $f : A \rightarrow R^n$, $f = (f_1, f_2, \dots, f_n)$ unde $f_i : A \rightarrow \mathbb{R}$.

Definiție. Se spune că funcția f este derivabilă în punctul $a \in A$ dacă toate funcțiile f_1, \dots, f_n sunt derivabile în a . Atunci derivata funcției f este

$$f'(a) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(a) & \dots & \frac{\partial f_1}{\partial x_m}(a) \\ \vdots & \dots & \vdots \\ \frac{\partial f_n}{\partial x_1}(a) & \dots & \frac{\partial f_n}{\partial x_m}(a) \end{pmatrix}.$$

Observație. Dacă toate funcțiile f_i admit derivate parțiale pe o vecinătate a punctului a , derivate continue în a , atunci f este derivabilă în punctul a .

Definiție. Se spune că funcția $f: A \subset \mathbb{R}^m \rightarrow \mathbb{R}^n$ este **de două ori derivabilă în punctul** a dacă f este derivabilă pe o bilă $B(a, r) \subset A$ iar $f': B(a, r) \rightarrow \mathbb{R}^n$ este derivabilă în punctul a .

Atunci:

$$f''(a) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(a) & \dots & \frac{\partial^2 f}{\partial x_m \partial x_1}(a) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_m}(a) & \dots & \frac{\partial^2 f}{\partial x_m^2}(a) \end{pmatrix}.$$

Teorema 6 (Schwarz). Dacă $f: A \subset \mathbb{R}^m \rightarrow \mathbb{R}^n$ este de două ori derivabilă în punctul a , atunci

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(a) = \frac{\partial^2 f}{\partial x_j \partial x_i}(a), \quad i, j \in \{1, \dots, m\}.$$

Teorema 7. Dacă f este de două ori derivabilă în punctul a , atunci

$$\lim_{x \rightarrow a} \frac{f(x) - f(a) - \sum_{i=1}^m \frac{\partial f}{\partial x_i}(a)(x_i - a_i) - \frac{1}{2} \sum_{i,j=1}^m \frac{\partial^2 f}{\partial x_i \partial x_j}(a)(x_j - a_j)(x_i - a_i)}{\|x - a\|^2} = 0.$$

XI.5. Puncte de extrem local

Definiție. Se spune că $a \in A \subset \mathbb{R}^m$ este **punct de maxim (minim) local** pentru funcția $f: A \rightarrow \mathbb{R}$, dacă există $r > 0$ astfel încât $f(x) \leq f(a)$, $\forall x \in B(a, r) \cap A$ ($f(x) \geq f(a)$, $\forall x \in B(a, r) \cap A$).

Dacă inegalitățile precedente sunt stricte pentru $x \neq a$, se spune că a este **punct de maxim (minim) local strict**.

Teorema 1 (Fermat). Fie $f: A \subset \mathbb{R}^m \rightarrow \mathbb{R}$ derivabilă în punctul de extrem local $x_0 \in A$. Dacă x_0 este punct interior mulțimii A , atunci $f'(x_0) = 0$,

$$f'(x_0) = 0 \Leftrightarrow \frac{\partial f}{\partial x_1}(x_0) = \dots = \frac{\partial f}{\partial x_m}(x_0) = 0.$$

Definiție. Un punct $x_0 \in A$ în care funcția f este derivabilă și $\frac{\partial f}{\partial x_1}(x_0) = \dots = \frac{\partial f}{\partial x_m}(x_0) = 0$ se numește **punct staționar (critic)** pentru funcția f .

Fie $f : A \subset \mathbb{R}^m \rightarrow \mathbb{R}$ o funcție de două ori derivabilă în punctul $a \in A$. Funcționala

$$H(h) = \sum_{i,j=1}^m \frac{\partial^2 f}{\partial x_i \partial x_j}(a) h_i h_j, \quad h = (h_1, \dots, h_m)$$

este o formă pătratică numită **hessiană** lui f în punctul a .

Definiție. Se spune că forma pătratică H este pozitiv definită (negativ definită) dacă $H(h) > 0$ pentru orice $h \in \mathbb{R}^m$, $h \neq 0$ ($H(h) < 0$, $\forall h \in \mathbb{R}^m \setminus \{0\}$).

Se spune că H este semipozitivă (seminegativă), dacă $H(h) \geq 0$, ($H(h) \leq 0$) $\forall h \in \mathbb{R}^m$.

Funcționala H se numește definită dacă este pozitiv definită sau negativ definită și se numește semidefinită dacă este semipozitivă sau seminegativă.

Teorema 2. Fie $f : A \subset \mathbb{R}^m \rightarrow \mathbb{R}$ de două ori derivabilă în punctul staționar x_0 .

- i) Dacă x_0 este punct de extrem local pentru f , atunci hessiană în x_0 este semidefinită.
- ii) Dacă hessiană în x_0 este strict pozitivă (negativă), atunci x_0 este punct de minim (maxim) local strict pentru f .

Se notează:

$$d_1 = \frac{\partial^2 f}{\partial x_1^2}(x_0),$$

$$d_2 = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2}(x_0) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(x_0) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(x_0) & \frac{\partial^2 f}{\partial x_2^2}(x_0) \end{vmatrix}, \dots,$$

$$d_m = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2}(x_0) & \dots & \frac{\partial^2 f}{\partial x_m \partial x_1}(x_0) \\ \vdots & \dots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_m}(x_0) & \dots & \frac{\partial^2 f}{\partial x_m^2}(x_0) \end{vmatrix}.$$

Conform teoremei Sylvester hessiană H este strict pozitivă dacă și numai dacă $d_0 > 0$, $\forall i \in \{1, \dots, m\}$ și strict negativă dacă și numai dacă $(-1)^i d_i > 0$, $\forall i \in \{1, \dots, m\}$. O afirmație analogă caracterizează hessiană semidefinită.

$$\text{În} \quad \text{cazul} \quad m = 2, \quad d_1 = \frac{\partial^2 f}{\partial x^2}(x_0),$$

$$d_2 = \Delta = \frac{\partial^2 f}{\partial x^2}(x_0) \frac{\partial^2 f}{\partial y^2}(x_0) - \left(\frac{\partial^2 f}{\partial x \partial y}(x_0) \right)^2.$$

Conform celor precedente, dacă x_0 este un punct staționar, adică $\frac{\partial f}{\partial x}(x_0) = \frac{\partial f}{\partial y}(x_0) = 0$, putem avea următoarele cazuri:

I. $\Delta > 0$ și atunci x_0 este punct de extrem și anume:

- dacă $\frac{\partial^2 f}{\partial x^2}(x_0) > 0$, atunci x_0 este punct de minim local strict;
- dacă $\frac{\partial^2 f}{\partial x^2}(x_0) < 0$, atunci x_0 este punct de maxim local strict.

II. $\Delta < 0$ și atunci x_0 nu este punct de extrem local.

III. $\Delta = 0$ caz în care nu se poate decide cu această teorie dacă x_0 este sau nu punct de extrem.

Teorema 3 (teorema funcțiilor implicate, cazul $m = 2$).

Fie $f : A \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ o funcție continuă astfel încât există $(x_0, y_0) \in A$,

$f(x_0, y_0) = 0$. Dacă f este derivabilă parțial în raport cu y , $\frac{\partial f}{\partial y} : A \rightarrow \mathbb{R}$ este

continuă și $\frac{\partial f}{\partial y}(x_0, y) \neq 0$ atunci există $\alpha, \beta > 0$, există și este unică o funcție

$\varphi : [x_0 - \alpha, x_0 + \alpha] \rightarrow [y_0 - \beta, y_0 + \beta]$ astfel încât $f(x, \varphi(x)) = 0$,
 $\forall x \in [x_0 - \alpha, x_0 + \alpha]$

Funcția φ este continuă, $\varphi(x_0) = y_0$. Dacă f este derivabilă în (x_0, y_0) , atunci φ este derivabilă în x_0 și

$$\varphi'(x_0) = -\frac{\frac{\partial f}{\partial x}(x_0, y_0)}{\frac{\partial f}{\partial y}(x_0, y_0)}.$$

XI.6. Integrala Riemann în \mathbb{R}^m

Integrala dublă

Fie $a \leq b, c \leq d$ și $(a, b) \times (c, d) \subset J \subset [a, b] \times [c, d]$. Numărul $\mu(J) = (d - c) \times (b - a)$ se numește **aria (măsura Jordan)** a lui J .

Fie $a < b, c < d$ și

$$a = x_0 < x_1 < \dots < x_n = b, c = y_0 < \dots < y_m = d.$$

$$\text{Fie } d = \left\{ [x_i, x_{i+1}] \times [y_j, y_{j+1}] \mid 0 \leq i \leq n-1, 0 \leq j \leq m-1 \right\}.$$

Familia d se numește **diviziune elementară** pentru dreptunghiul I .

În d se află $p = n \cdot m$ elemente ceea ce se sintetizează prin $d = \{I_1, \dots, I_p\}$. Mai general, orice sistem $\{J_1, \dots, J_p\}$ astfel încât $\text{int } I_k \subset J_k \subset I_k, \bigcup_{k=1}^p J_k = I$ se numește, de asemenea, **diviziune elementară** pentru I . Evident

$$\mu(I) = \sum_{k=1}^p \mu(J_k).$$

Se notează cu Δ familia tuturor diviziunilor elementare ale lui I .

Fie $f : I \rightarrow \mathbb{R}$ o funcție mărginită, $\delta = \{I_1, \dots, I_p\}$ o diviziune elementară

a lui I și $M_k = \sup_{t \in I_k} f(t), m_k = \inf_{t \in I_k} f(t)$. Sumele $S_\delta(f) = \sum_{k=1}^p M_k \mu(I_k)$,

$s_\delta(f) = \sum_{k=1}^p m_k \mu(I_k)$ se numesc **suma Darboux superioară**, respectiv **inferioară**,

asociată funcției f și diviziunii δ .

Evident

$$\mu(t) \inf_{t \in I} f(t) \leq s_\delta(f) \leq S_\delta(f) \leq \mu(t) \sup_{t \in I} f(t).$$

Numerele

$$\sup_{d \in \Delta} s_d(f) = \underline{\iint} f(x, y) dx dy$$

$$\inf_{d \in \Delta} S_d(f) = \overline{\iint} f(x, y) dx dy$$

se numesc **integrala inferioară**, respectiv **superioară**, a funcției f pe dreptunghiul I .

Definiție. Funcția mărginită $f : I \rightarrow \mathbb{R}$ se numește **integrabilă Riemann** dacă $\underline{\iint_I} f(x, y) dx dy = \overline{\iint_I} f(x, y) dx dy$. Se notează atunci

$$\underline{\iint_I} f(x, y) = \overline{\iint_I} f(x, y)$$

și se numește **integrala Riemann** a funcției f pe dreptunghiul I .

Fie $d_1, d_2 \in \Delta$.

Definiție. Se spune că d_2 este **mai fină** decât d_1 dacă orice dreptunghi din d_1 este o reuniune de dreptunghiuri din d_2 . Se notează $d_1 \leq d_2$.

Observație. Dacă $d_1 \leq d_2$ și $f : I \rightarrow \mathbb{R}$ este mărginită, atunci

$$s_{d_1}(f) \leq s_{d_2}(f) \leq S_{d_2}(f) \leq S_{d_1}(f).$$

Rezultă că pentru orice diviziuni δ', δ'' are loc $s_{\delta'}(f) \leq S_{\delta''}(f)$ iar de aici că

$$\underline{\iint_I} f(x, y) dx dy \leq \overline{\iint_I} f(x, y) dx dy.$$

Teorema 1 (Criteriul lui Darboux). Fie $f : I \rightarrow \mathbb{R}$ o funcție mărginită. Atunci f este integrabilă Riemann dacă și numai dacă pentru orice $\varepsilon > 0$ există $d \in \Delta$ astfel încât $S_d(f) - s_d(f) < \varepsilon$.

Corolar. Orice funcție continuă $f : I \rightarrow \mathbb{R}$ este integrabilă Riemann.

Se notează $\mathcal{R}(I)$ mulțimea tuturor funcțiilor integrabile Riemann pe I .

Fie $d \in \Delta$, $d = \{I_1, \dots, I_p\}$, $f : I \rightarrow \mathbb{R}$, $\xi_j \in I_j$. Numărul

$$\sigma_d(f, \xi) = \sum_{j=1}^p f(\xi_j) \mu(I_j)$$

se numește **sumă Riemann** asociată funcției f , diviziunii d și sistemului de puncte $\{\xi_1, \dots, \xi_p\}$.

Teorema 2 (criteriul cu sume Riemann după finețe). Funcția $f : I \rightarrow \mathbb{R}$ este integrabilă Riemann dacă și numai dacă este mărginită și există $\lambda \in \mathbb{R}$ și pentru orice $\varepsilon > 0$ există $d_0 \in \Delta$ astfel încât pentru orice $d \in \Delta$, $d \geq d_0$ avem $|\sigma_d(f, \xi) - \lambda| < \varepsilon$. Atunci

$$\lambda = \underline{\iint_I} f(x, y) dx dy.$$

Pentru orice dreptunghi J , $\text{diam } J$ este **lungimea diagonalei** lui J .

Dacă $d = \{I_1, \dots, I_p\}$ este o diviziune a lui I , atunci numărul

$$\|d\| = \max_{1 \leq k \leq p} \text{diam } I_k$$
 se numește **norma diviziunii** d .

Teorema 3 (criteriul cu sume Riemann). Funcția $f : I \rightarrow \mathbb{R}$ este integrabilă Riemann dacă și numai dacă există $\lambda \in \mathbb{R}$ și pentru orice $\varepsilon > 0$, există $\eta > 0$, astfel încât pentru orice sumă Riemann $\sigma_d(f, \xi)$, cu $\|d\| < \eta$ avem $|\sigma_d(f, \xi) - \lambda| < \varepsilon$. Atunci

$$\lambda = \iint_I f(x, y) dx dy.$$

Corolar. Funcția $f : I \rightarrow \mathbb{R}$ este integrabilă Riemann dacă și numai dacă există $\lambda \in \mathbb{R}$ și pentru orice sir de sume Riemann $(\sigma_{d_n}(f, \xi))_{n \in \mathbb{N}}$, cu $\lim_{n \rightarrow \infty} \|d_n\| = 0$, avem $\lim_{n \rightarrow \infty} \sigma_{d_n}(f, \xi) = \lambda$. Atunci

$$\lambda = \iint_I f(x, y) dx dy.$$

Definiție. Mulțimea $A \subset \mathbb{R}^2$ se numește **neglijabilă Lebesgue** dacă pentru orice $\varepsilon > 0$, există $\{I_n | n \in \mathbb{N}\}$ o familie de dreptunghiuri astfel încât $A \subset \bigcup_{n \in \mathbb{N}} I_n$

$$\text{și } \sum_{n=1}^{\infty} \mu(I_n) < \varepsilon.$$

Observații

1. Dacă A este neglijabilă și $B \subset A$, atunci B este neglijabilă.
2. Orice reuniune numărabilă de mulțimi neglijabile este o mulțime neglijabilă.
3. Dacă I este un dreptunghi, atunci ∂I este o mulțime neglijabilă.

Teorema 4 (criteriul lui Lebesgue). Funcția $f : I \rightarrow \mathbb{R}$ este integrabilă Riemann dacă și numai dacă este mărginită și mulțimea punctelor de discontinuitate este neglijabilă.

Teorema 4. Fie $f : [a, b] \times [c, d] \rightarrow \mathbb{R}$ integrabilă astfel încât $\forall y \in [c, d]$ funcția $x \mapsto f(x, y)$ este integrabilă pe $[a, b]$. Atunci funcția $\varphi(y) = \int_a^b f(x, y) dx$ este integrabilă pe $[c, d]$ și

$$\iint_{[a, b] \times [c, d]} f(x, y) dx dy = \int_c^d \left(\int_a^b f(x, y) dx \right) dy.$$

Corolar. Dacă $f : [a, b] \times [c, d] \rightarrow \mathbb{R}$ este continuă, atunci

$$\iint_{[a, b] \times [c, d]} f(x, y) dx dy = \int_c^d \left(\int_a^b f(x, y) dx \right) dy = \int_a^b \left(\int_c^d f(x, y) dy \right) dx.$$

Definiție. Mulțimea mărginită $E \subset \mathbb{R}^2$ se numește **măsurabilă Jordan** dacă ∂E este neglijabilă Lebesgue.

Orice dreptunghi este o mulțime măsurabilă Jordan.

Teorema 5. Fie I un dreptunghi închis și $E \subset I$. Funcția caracteristică $\chi_E : I \rightarrow \mathbb{R}$ este integrabilă Riemann dacă și numai dacă E este măsurabilă Jordan.

Numărul $\mu(E) = \int_I \chi_E(x) dx$ se numește **măsura Jordan** a lui E (se notează și $\mu(E) = \int_E dx$).

Dacă E este un dreptunghi atunci măsura sa Jordan coincide cu aria sa.

Teorema 6. Fie A, B mulțimi măsurabile Jordan. Atunci:

1. $A \cup B, A \cap B, A \setminus B, \overset{\circ}{A}$ și \bar{A} sunt măsurabile Jordan.
2. $A \cap B = \emptyset \Rightarrow \mu(A \cup B) = \mu(A) + \mu(B)$.
3. $A \subset B \Rightarrow \mu(B \setminus A) = \mu(B) - \mu(A); \mu(A) \leq \mu(B)$.
4. $\mu(A \cup B) \leq \mu(A) + \mu(B)$.
5. Dacă E este măsurabilă Jordan și neglijabilă, atunci $\mu(E) = 0$.
6. $A \cap B$ neglijabilă, atunci $\mu(A \cup B) = \mu(A) + \mu(B)$.

Definiție. Mulțimea A se numește **elementară** dacă este o reuniune finită de dreptunghiuri, două câte două având intersecția neglijabilă.

Teorema 7. Mulțimea mărginită E este măsurabilă Jordan dacă și numai dacă pentru orice $\varepsilon > 0$ există A, B mulțimi elementare astfel încât $A \subset E \subset B$ și $\mu(B) - \mu(A) < \varepsilon$.

Propoziție. Dacă A este măsurabilă Jordan și $z \in \mathbb{R}^2$ atunci $A + z$ este măsurabilă Jordan și $\mu(A + z) = \mu(A)$.

Observație. Dacă E este măsurabilă Jordan, atunci E este neglijabilă dacă și numai dacă $\text{int } E = \emptyset$.

Fie $E \subset \mathbb{R}^2$ o mulțime mărginită.

Definiție. Se spune că funcția $f : E \rightarrow \mathbb{R}$ este **integrabilă** dacă există un dreptunghi D astfel încât $E \subset D$ și funcția

$$g(x) = \begin{cases} f(x) & \text{dacă } x \in E \\ 0 & \text{dacă } x \in D \setminus E \end{cases}$$

este integrabilă. Atunci $\int_E f(x) dx = \int_D f(x) dx$.

Fie $E \subset \mathbb{R}^2$, mărginită, măsurabilă Jordan. Se notează

$$\mathcal{R}(E) = \{f : E \rightarrow \mathbb{R} \mid f \text{ integrabilă Riemann}\}.$$

Teorema 8 (critriul lui Lebesgue). $f \in \mathcal{R}(E)$ dacă și numai dacă f este mărginită și mulțimea punctelor de discontinuitate este neglijabilă.

Teorema 9 (proprietăți ale integralei). Fie $E \subset \mathbb{R}^2$ mărginită, măsurabilă Jordan.

1. Dacă $f \in \mathcal{R}(E)$ și $A \subset E$, A măsurabilă Jordan, atunci $f|_A \in \mathcal{R}(E)$.

2. Dacă $f, g \in \mathcal{R}(E)$ și $\alpha, \beta \in \mathbb{R}$ atunci $\alpha f + \beta g \in \mathcal{R}(E)$ și

$$\iint_E (\alpha f(x, y) + \beta g(x, y)) dx dy = \alpha \iint_E f(x, y) dx dy + \beta \iint_E g(x, y) dx dy;$$

$$f \leq g \Rightarrow \iint_E f(x, y) dx dy \leq \iint_E g(x, y) dx dy;$$

$$f \cdot g \in \mathcal{R}(E).$$

3. Dacă $f \in \mathcal{R}(E)$ atunci $|f| \in \mathcal{R}(E)$ și

$$\left| \iint_E f(x, y) dx dy \right| \leq \iint_E |f(x, y)| dx dy.$$

4. Dacă $f \in \mathcal{R}(E)$, $E = E_1 \cup E_2$, E_1, E_2 măsurabile Jordan, $E_1 \cap E_2$ neglijabilă, atunci

$$\iint_E f(x, y) dx dy = \iint_{E_1} f(x, y) dx dy + \iint_{E_2} f(x, y) dx dy.$$

5. Dacă $(f_n)_{n \in \mathbb{N}}$ este un sir din $\mathcal{R}(E)$, convergent uniform către funcția f , atunci $f \in \mathcal{R}(E)$ și

$$\lim_{n \rightarrow \infty} \iint_E f_n(x, y) dx dy = \iint_E f(x, y) dx dy.$$

Definiție. Mulțimea $E \subset \mathbb{R}^2$ se numește **simplă în raport cu axa Oy** dacă există $\varphi, \psi : [a, b] \rightarrow \mathbb{R}$, funcții continue, $\varphi \leq \psi$ astfel încât

$$E = \{(x, y) \mid x \in [a, b], \varphi(x) \leq y \leq \psi(x)\}.$$

Analog, Mulțimea E se numește **simplă în raport cu axa Ox** dacă există $\varphi, \psi : [c, d] \rightarrow \mathbb{R}$, funcții continue, $\varphi \leq \psi$ astfel încât

$$E = \{(x, y) \mid y \in [c, d], \varphi(y) \leq x \leq \psi(y)\}.$$

Orice mulțime simplă în raport cu una din axe este compactă, măsurabilă Jordan.

Teorema 10 (formula de integrare iterată pentru domenii simple). Fie $E \subset \mathbb{R}^2$ un domeniu simplu în raport cu axa Oy și $f : E \rightarrow \mathbb{R}$ o funcție continuă. Atunci

$$\iint_E f(x, y) dx dy = \int_a^b \left(\int_{\varphi(x)}^{\psi(x)} f(x, y) dy \right) dx.$$

O formulă analoagă are loc în cazul în care E este simplu în raport cu axa Ox :

$$\iint_E f(x, y) dx dy = \int_c^d \left(\int_{\varphi(y)}^{\psi(y)} f(x, y) dx \right) dy.$$

Teorema 11 (formula de schimbare de variabilă). Fie $A \subset \mathbb{R}^2$ o mulțime deschisă, $T: A \rightarrow \mathbb{R}^2$ o funcție derivabilă cu derivata continuă, $E \subset A$, E compactă, măsurabilă Jordan astfel încât T să fie injectivă pe $\text{int } E$ și $\det T'(t) \neq 0$ pentru orice $t \in A$. Fie $f: T(E) \rightarrow \mathbb{R}$ o funcție continuă. Atunci

$$\iint_{T(E)} f(x, y) dx dy = \iint_E (f \circ T)(u, v) |\det T'(u, v)| du dv.$$

Formula lui Green

Fie $E = [0,1] \times [0,1]$ și $\lambda_1(t) = (t, 0)$, $\lambda_2(t) = (1, t)$, $\lambda_3(t) = (1-t, 1)$, $\lambda_4(t) = (0, 1-t)$, $t \in [0,1]$. Fie $\partial E = \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_4$.

Teorema 12. Fie $A \subset \mathbb{R}^2$, A deschisă, astfel încât $E \subset A$. Fie $\varphi: A \rightarrow \mathbb{R}$ o funcție de două ori derivabilă astfel încât este injectivă pe interiorul mulțimii E și $\det \varphi'(t) > 0$ pentru orice $t \in \text{int } E$. Fie D o mulțime deschisă astfel încât $\varphi(E) \subset D$ și $P, Q: D \rightarrow \mathbb{R}$ derivabile, cu derivatele continue. Atunci

$$\iint_{\varphi(E)} \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dx dy = \oint_{\varphi(\partial E)} P dx + Q dy.$$

Observație. Dacă M este un domeniu simplu în raport cu una din axe iar ∂M este un drum simplu, închis, de clasă C^1 pe porțiuni, a cărui imagine este frontiera topologică a lui M și astfel încât sensul de parcursere pe ∂M lasă domeniul în stânga (∂M este pozitiv orientat), atunci formula precedentă este

$$\iint_M \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dx dy = \oint_{\partial M} P dx + Q dy.$$

Formula precedentă este adevărată în condiții mai generale, pentru domenii mărginite de imaginea unui drum simplu, închis, de clasă C^1 pe porțiuni, pozitiv orientat. În particular, un asemenea domeniu este măsurabil Jordan iar aria sa este atunci

$$\mu(M) = \frac{1}{2} \oint_{\partial M} x dy - y dx.$$

Observație. Se mai spune în formulele precedente că integrala curbilinie se face de-a lungul curbei în sens direct.

Integrala triplă

Construcția integralei Riemann în \mathbb{R}^3 este analoagă celei duble. Astfel, o mulțime de forma $[a,b] \times [c,d] \times [e,f]$ (paralelipiped dreptunghic) este mulțimea de bază de la care se pornește construcția integralei triple. **Măsură Jordan (volumul)** lui I este $\mu(I) = (b-a)(d-c)(f-e)$.

Definiție. Mulțimea $A \subset \mathbb{R}^3$ se numește **neglijabilă Lebesgue** dacă $\forall \varepsilon > 0$ există $(I_n)_{n \in \mathbb{N}}$ un sir de paralipipede astfel încât $A \subset \bigcup_{n=0}^{\infty} I_n$ și $\sum_{n=0}^{\infty} \mu(I_n) < \varepsilon$.

Definiție. Mulțimea mărginită $E \subset \mathbb{R}^3$ se numește **măsurabilă Jordan** dacă frontieră sa este neglijabilă Lebesgue.

Definiția integrabilității Riemann este analogă celei din \mathbb{R}^2 .

Orice funcție continuă este integrabilă pe orice mulțime compactă măsurabilă Jordan.

Mai general:

Teorema 1 (criteriul lui Lebesgue). Fie $E \subset \mathbb{R}^3$ o mulțime compactă măsurabilă Jordan. Funcția $f : E \rightarrow \mathbb{R}$ este integrabilă Riemann dacă și numai dacă este mărginită și mulțimea punctelor ei de discontinuitate este neglijabilă.

Se notează

$$\iiint_D f(x, y, z) dx dy dz.$$

Pentru un domeniu compact, măsurabil Jordan E , măsura sa Jordan (volumul) este

$$\mu(E) = \iiint_E dx dy dz.$$

Proprietățile integralei și ale măsurii sunt aceleași ca la integrala dublă.

Definiție. Mulțimea $E \subset \mathbb{R}^3$ se numește **simplă în raport cu axa Oz** dacă există $D \subset \mathbb{R}^2 (= \mathbb{R}^2 \times \{0\})$ o mulțime compactă măsurabilă Jordan și $\varphi, \psi : D \rightarrow \mathbb{R}$ funcții continue, $\varphi \leq \psi$ și

$$E = \{(x, y, z) | (x, y) \in D, \varphi(x, y) \leq z \leq \psi(x, y)\}.$$

Analog se definește noțiunea de domeniu simplu în raport cu celelalte axe.

Orice domeniu simplu în raport cu una din axe este compact, măsurabil Jordan.

Teorema 2. Fie $E \subset \mathbb{R}^3$ o mulțime simplă în raport cu axa Oz (ca în definiția precedentă) și $f : E \rightarrow \mathbb{R}$ o funcție continuă. Atunci

$$\iiint_E f(x, y, z) dx dy dz = \iint_D \left(\int_{\varphi(x, y)}^{\psi(x, y)} f(x, y, z) dz \right) dx dy.$$

Formule analoage au loc pentru domenii simple în raport cu celelalte axe.

Formula de schimbare de variabilă are același enunț ca în cazul integralei duble.

În particular transformarea următoare, T , numită **trecerea în coordonate sféricе** este

$$\begin{cases} x = \rho \cos \theta \cos \varphi, \\ y = \rho \sin \theta \cos \varphi, \\ z = \rho \sin \varphi \end{cases}$$

unde $\rho = \sqrt{x^2 + y^2 + z^2}$, θ este măsura unghiului făcut de direcția pozitivă a axei Ox cu proiecția ON a lui OM pe planul xOy iar φ este măsura unghiului făcut de ON cu OM considerat pozitiv în semispațiul $z \geq 0$ și negativ dacă $z \leq 0$ (toate acestea pentru $M \neq 0$).

Jacobianul transformării este $\rho^2 \cos \varphi$ iar formula de schimbare de variabilă este:

$$\iiint_M f(x, y, z) dx dy dz = \iiint_{T^{-1}(M)} f(\rho \cos \theta \cos \varphi, \rho \sin \theta \cos \varphi, \rho \sin \varphi) \rho^2 \cos \varphi d\rho d\theta d\varphi.$$

BIBLIOGRAFIE

1. Boboc N., *Analiză matematică*, vol I, II, Editura Universității București, 1998.
2. Siretchi Gh., *Calcul diferențial și integral*, Editura Științifică și Enciclopedică, București, 1985.
3. Duda I., Trandafir R., *Analiză matematică I, Exerciții și probleme*, Editura Fundației România de Mâine, 2002.
4. Trandafir R., Duda I., *Analiză matematică*, partea a II-a, multiplicare Departamentalul ID, Universitatea Spiru Haret, 2002.

GEOMETRIE ANALITICĂ

Prof. univ. dr. GABRIEL PRIPOAIE

I. SPAȚII VECTORIALE; DEFINIȚII; EXEMPLE

Definiție. Fie K un corp comutativ. Fie o mulțime $V \neq \emptyset$ și două funcții:

$$+: V \times V \rightarrow V$$

$$\cdot: K \times V \rightarrow V$$

Atunci tripletul $(V, +, \cdot)$ se numește spațiu vectorial peste corpul K (sau K -spațiu vectorial), dacă:

1. $(V, +)$ este grup abelian;
2. $a(v + w) = av + aw$, pentru $\forall a \in K, \forall v, w \in V$;
3. $(a + b)v = av + bv$, pentru $\forall a, b \in K, \forall v \in V$;
4. $a(bv) = (ab)v$, pentru $\forall a, b \in K, \forall v \in V$;
5. $1 \cdot v = v$, pentru $\forall v \in V$;

Observații.

- (i) Dacă $K = \mathbb{R}$, atunci V se numește spațiu vectorial real.
- (ii) Dacă $K = \mathbb{C}$, atunci V se numește spațiu vectorial complex.

Propoziție. Fie V -K spațiu vectorial. Atunci:

- (i) $0 \cdot v = 0, \forall v \in V \forall$;
- (ii) $a \cdot 0 = 0, \forall a \in K$;
- (iii) $av = 0 \Leftrightarrow a = 0$ sau $v = 0$.

Exemple.

1. Orice corp K comutativ este spațiu vectorial peste el însuși în raport cu operația de adunare, iar legea externă este înmulțirea obișnuită din K .

Observație. $(\mathbb{R}, +, \cdot)$ este spațiu vectorial real; $(\mathbb{C}, +, \cdot)$ este spațiu vectorial complex.

2. Fie K un corp comutativ și

$$K^n = K \times K \times \dots \times K = \{(x_1, x_2, \dots, x_n) | x_i \in K, i = \overline{1, n}\}.$$

Pe K^n definim legile de compozиție:

$$+: K^n \times K^n \rightarrow K^n,$$

$$(x_1, x_2, \dots, x_n) + (y_1, y_2, \dots, y_n) = (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n),$$

$$\forall x, y \in K^n, x = (x_1, x_2, \dots, x_n), y = (y_1, y_2, \dots, y_n)$$

și

$$\cdot : K^n \times K^n \rightarrow K^n, \quad a(x_1, x_2, \dots, x_n) = (ax_1, ax_2, \dots, ax_n), \quad \forall a \in K, \\ x = (x_1, x_2, \dots, x_n) \in K^n.$$

Atunci $(K^n, +, \cdot)$ este spațiu vectorial peste corpul K .

Observație. $(\mathbb{R}^n, +, \cdot)$ este spațiu vectorial peste \mathbb{R} ; $(\mathbb{C}^n, +, \cdot)$ este spațiu vectorial peste \mathbb{C} .

II. MORFISME DE SPAȚII VECTORIALE

Fie V și W două K -spații vectoriale, iar $f : V \rightarrow W$.

Definiție. f se numește **morfism** de K -spații vectoriale dacă

- a) $f(x+y) = f(x) + f(y)$, $\forall x, y \in V$;
- b) $f(ax) = af(x)$, $\forall a \in K$, $\forall x \in V$.

Observații.

1. Dacă în plus f este bijectivă, atunci f este **izomorfism** de K -spații vectoriale.
2. Un morfism de la V la V se numește **endomorfism**. Multimea endomorfismelor lui V se notează $\text{End}_K(V)$.
3. Un izomorfism de la V la V se numește **automorfism**. Multimea automorfismelor lui V se notează $\text{Aut}_K(V)$.
4. Multimea $\text{Im } f = \{f(x) \in W, x \in V\}$ se numește **imaginea** lui f .
5. Multimea $\text{Ker } f = \{x \in V | f(x) = 0_W\}$ se numește **nucleul** lui f .

Propoziție. Fie V și W două K -spații vectoriale și $f : V \rightarrow W$ o funcție. Atunci f este morfism de K -spații vectoriale dacă și numai dacă $f(ax+by) = af(x) + bf(y)$, $\forall x, y \in V$, $\forall a, b \in K$.

Exemple.

1. Fie K^n spațiu vectorial peste K . Funcția $p_i : K^n \rightarrow K$, $\forall i = \overline{1, n}$, $p_i(a_1, \dots, a_n) = a_i$ este morfism.
2. $f : \mathbf{M}_{m,n}(\mathbb{R}) \rightarrow \mathbf{M}_{m,n}(\mathbb{R})$, $f(A) = A^t$ este morfism de spații vectoriale.

III. TEOREME DE CLASIFICARE PENTRU SPAȚII VECTORIALE FINIT DIMENSIONALE

III.1. Vectori liniari independenți, sistem de generatori, bază dimensiune

Fie V un K -spațiu vectorial și o mulțime finită S inclusă în V , $S = \{e_1, e_2, \dots, e_n\}$, numit sistem de vectori din V .

Definiție. S se numește **sistem liniar independent** dacă din orice egalitate $a^1 e_1 + a^2 e_2 + \dots + a^n e_n = 0$ avem $a^1 = a^2 = \dots = a^n = 0$.

Observație. În acest caz spunem că vectorii e_1, \dots, e_n sunt **liniar independenți**. În caz contrar S se numește **sistem liniar dependent**, iar vectorii e_1, \dots, e_n se numesc **liniar dependenți**.

Definiție. Dacă un vector $x \in V$ se poate scrie sub forma $x = \alpha^1 e_1 + \dots + \alpha^n e_n$, atunci spunem că x este **combinație liniară** a vectorilor e_1, \dots, e_n cu scalari în corpul K .

Definiție. S se numește **sistem de generatori** al lui V dacă oricare ar fi $x \in V$, x se poate scrie ca o combinație liniară a vectorilor din S .

Mai spunem că S generează spațiul V .

Definiție. S se numește **bază** a lui V dacă S este simultan **sistem de generatori** al lui V și **sistem liniar independent**.

Proprietăți.

- (i) Dacă $S = \{e_1, \dots, e_n\}$ este un sistem de vectori liniari independenți, atunci S nu conține vectorul nul.
- (ii) Dacă S este un sistem liniar independent și $S' \subseteq S$ un subsistem nevid, atunci S' este tot liniar independent.
- (iii) Dacă $S = \{e_1\}$, atunci S este un sistem liniar independent dacă și numai dacă e_1 este nenul.
- (iv) Fie $S = \{v, w\}$ cu $v \neq w$. Atunci S este un sistem liniar independent dacă v și w nu sunt coliniari.

Observație. Noțiunile de sistem liniar independent și de sistem de generatori se extind la sistemele de vectori infinite.

Teorema bazei. Fie V un K -spațiu vectorial și $S = \{e_1, \dots, e_n\}$ un sistem de generatori pentru spațiul V . Fie $S' \subseteq S$ un sistem de vectori liniari independenți. Atunci există $B \subseteq V$ bază pentru V astfel încât $S' \subseteq B \subseteq S$.

Corolar.

- i) Din orice sistem de generatori se poate extrage o bază.
- ii) Orice bază poate fi completată până la un sistem de generatori (sau până la o bază).

iii) *Orice spațiu vectorial $V \neq \{0\}$ admite o bază.*

Definiție. Fie V un spațiu vectorial peste corpul K și $B = \{e_1, \dots, e_n\}$ o bază a lui V . Atunci n se numește **dimensiunea spațiului vectorial V** .

Observații.

- 1) Notăm dimensiunea spațiului vectorial V peste corpul K prin $\dim_K V = n$;
- 2) Dacă B este mulțime infinită, atunci notăm $\dim_K V = \infty$.

Lema schimbului. Fie V un spațiu vectorial peste corpul K , $B = \{e_1, \dots, e_n\}$ o bază a lui V și fie w un vector de forma $w = a^1 e_1 + \dots + a^n e_n$; $a^{i_0} \neq 0$; $a^1, \dots, a^n \in K$. Atunci $\bar{B} = \{e_1, \dots, e_{i_0-1}, w, e_{i_0+1}, \dots, e_n\}$ este tot bază a lui V .

Exemple:

- 1) Dacă avem doi vectori $v_1 = (-1, 2, 2)$, $v_2 = (3, 0, -2)$ și $\lambda_1 = 2$, $\lambda_2 = 4$, atunci $v = \lambda_1 v_1 + \lambda_2 v_2 = 2(-1, 2, 2) + 4(3, 0, -2) = (10, 4, -4)$, adică v este combinația liniară a vectorilor v_1 și v_2 .
- 2) În \mathbb{R}^3 , sistemul $S = \{(1, 0, 0), (0, 3, 0)\}$ este liniar independent.

Observație. Sistemul $B = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ este bază în \mathbb{R}^3 .

Observație. În mod analog, în spațiu vectorial K^n , sistemul

$$B = \{e_1, \dots, e_n\} \text{ este bază canonică, unde } \begin{cases} e_1 = (1, 0, 0, \dots, 0) \\ e_2 = (0, 1, 0, \dots, 0) \\ \dots \\ e_n = (0, 0, \dots, 0, 1) \end{cases}$$

III.2. Teoreme de clasificare

Teorema Schimbului (Steintz). Fie V un spațiu vectorial peste corpul K , $B = \{e_1, \dots, e_n\}$ o bază a lui V , $S = \{f_1, \dots, f_p\}$ un sistem de vectori liniar independenți. Atunci

- (i) $p \leq n$;
- (ii) $\{f_1, \dots, f_p, e_{p+1}, \dots, e_n\}$ este o bază a lui V (modulo o renumerare a indicilor).

Corolar. Orice două baze ale lui V au același număr de elemente.

Teoremă. Fie V și W spații vectoriale finit dimensionale peste corpul K .

- (i) Dacă $\dim_K V = \dim_K W$, atunci V și W sunt izomorfe.
- (ii) Dacă V și W sunt izomorfe, atunci $\dim_K V = \dim_K W$.

Corolar. Fie V un spațiu vectorial peste corpul K și $\dim_K V = n$. Atunci V este izomorf cu K^n . În particular, orice spațiu real n -dimensional este izomorf cu \mathbb{R}^n și orice spațiu vectorial complex n -dimensional este izomorf cu \mathbb{C}^n .

Observație. Fie V un spațiu vectorial peste corpul K , $B = \{e_1, \dots, e_n\}$ o bază a lui V . Un vector x din V se scrie sub forma $x = x^1 e_1 + \dots + x^n e_n$, unde $x^1, \dots, x^n \in K$. Vectorul x se mai scrie (x^1, \dots, x^n) ; x^1, \dots, x^n se numesc coordonatele vectorului x în baza B .

III.3. Schimbarea de bază și de coordonate

Fie V un spațiu vectorial peste corpul K , $\dim_K V = n$, $B = \{e_1, \dots, e_n\}$ și $\bar{B} = \{\bar{e}_1, \dots, \bar{e}_n\}$ baze ale lui V . Dacă $x \in V$, atunci $x = (x^1, \dots, x^n)$ se numește

vector linie, iar $x = \begin{pmatrix} x^1 \\ \vdots \\ x^n \end{pmatrix}$ se numește vector coloană. Scriind vectorii bazei B ca

vectori coloană putem forma o matrice asociată bazei B ,

$$B = \begin{pmatrix} e_1^1 & e_2^1 & \cdots & e_n^1 \\ e_1^2 & e_2^2 & \cdots & e_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ e_1^n & e_2^n & \cdots & e_n^n \end{pmatrix} \in M_{n,n}(K)$$

și similar pentru \bar{B} , cu $\det B \neq 0$ și $\det \bar{B} \neq 0$ pentru că vectorii bazelor sunt liniar independenți. Putem scrie

$$B = \begin{pmatrix} \bar{e}_1^1 & \bar{e}_2^1 & \cdots & \bar{e}_n^1 \\ \bar{e}_1^2 & \bar{e}_2^2 & \cdots & \bar{e}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{e}_1^n & \bar{e}_2^n & \cdots & \bar{e}_n^n \end{pmatrix} = \begin{pmatrix} e_1^1 & e_2^1 & \cdots & e_n^1 \\ e_1^2 & e_2^2 & \cdots & e_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ e_1^n & e_2^n & \cdots & e_n^n \end{pmatrix} \begin{pmatrix} p_1^1 & p_2^1 & \cdots & p_n^1 \\ p_1^2 & p_2^2 & \cdots & p_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ p_1^n & p_2^n & \cdots & p_n^n \end{pmatrix}.$$

Astfel $\bar{B} = BP$.

Notăție. $GL(n, K) = \{P \in M_{n,n}(K) | \det P \neq 0\}$.

Observații.

- 1) Fie B o bază fixată a lui V , $\dim_K V = n$. Atunci orice altă bază a lui V se poate obține prin înmulțirea lui B cu o matrice din $GL(n, K)$.
- 2) P se numește matricea de trecere de la baza B la baza \bar{B} .

- 3) Schimbarea de coordonate se face tot prin intermediul matricei de trecere P după regula $x = P\bar{x}$.

Exemplu.

- 1) Fie vectorul $x = (1, 2, 3) \in \mathbb{R}^3$ în baza canonică $B = \{e_1 = (1, 0, 0), e_2 = (0, 1, 0), e_3 = (0, 0, 1)\}$. Să se determine coordonatele vectorului x în baza $\bar{B} = \{\bar{e}_1 = (1, 1, 1), \bar{e}_2 = (1, 2, 0), \bar{e}_3 = (3, 0, 0)\}$.

- 2) În \mathbb{R}^3 considerăm sistemele de vectori

$$B' = \{e'_1 = (1, 2, 1), e'_2 = (2, 3, 3), e'_3 = (3, 7, 1)\}$$

și

$$B'' = \{e''_1 = (3, 1, 4), e''_2 = (5, 2, 1), e''_3 = (1, 1, -6)\}.$$

Să se arate că B' și B'' sunt baze. Să se găsească matricele S' și S'' de trecere de la baza canonică B din \mathbb{R}^3 la B' și B'' și să se deducă de aici matricea de trecere de la B' la B'' .

IV. SUBSPAȚII VECTORIALE

Definiție. Fie V un spațiu vectorial peste corpul K și W o submulțime nevidă a lui V . W se numește **subspațiu vectorial** al lui V dacă cele două operații restricționate la W ,

$$\begin{aligned} +|_{W \times W} : W \times W &\rightarrow W \\ \cdot|_{K \times W} : K \times W &\rightarrow W \end{aligned}$$

determină pe W o structură de K -spațiu vectorial.

Observații.

- 1) W este spațiu vectorial peste corpul K dacă W împreună cu operațiile din V determină o structură de spațiu vectorial peste corpul K .
- 2) W este subspațiu vectorial al lui V dacă $\forall w_1, w_2 \in W, w_1 + w_2 \in W$ și $\forall a \in K, \forall w \in W, aw \in W$.

Propoziție. $W \subset V$ este subspațiu vectorial peste corpul K dacă și numai dacă oricare ar fi $w_1, w_2 \in W$ și oricare ar fi $a^1, a^2 \in K$ avem $a^1 w_1 + a^2 w_2 \in W$.

Exemplu. Fie V un spațiu vectorial peste corpul K , atunci $W_1 = \{0_V\}$, care se numește subspațiu nul și $W_2 = V$, numit subspațiu total, sunt subspații improprii ale lui V .

Definiție. Fie V un spațiu vectorial peste corpul K și $S = \{e_1, \dots, e_n\}$. Se numește **subspațiu generat de S** , mulțimea tuturor vectorilor care se pot scrie ca o combinație liniară a vectorilor lui S . Notăm:

$$\bar{S} = \langle S \rangle = sp\{e_1, \dots, e_n\} = \left\{ a^1 e_1 + \dots + a^n e_n \mid a^1, \dots, a^n \in K \right\}.$$

În particular, dacă S este sistem de generatori pentru V atunci $\bar{S} = V$.

Exemplu. Mulțimea S a tuturor soluțiilor $x \in K^n$ ale sistemului

$$\begin{cases} a^1 x^1 + \dots + a_n^1 x^n = 0 \\ a^2 x^1 + \dots + a_n^2 x^n = 0 \\ \dots \\ a^m x^1 + \dots + a_n^m x^n = 0 \end{cases}, \text{ unde } A = (a_{\alpha}^i) \in M_{m,n}(K), \quad i = \overline{1, m}, \quad \alpha = \overline{1, n}, \text{ este}$$

subspațiu vectorial al lui K^n .

Propoziție. Fie V și W spații vectoriale peste corpul K și $f : V \rightarrow W$ morfism de spații vectoriale.

- (i) Dacă $V_1 \subset V$ este subspațiu vectorial al lui V , atunci $f(V_1)$ este subspațiu vectorial al lui W . În particular, $\text{Im } f = f(V)$ este subspațiu vectorial al lui W .
- (ii) Dacă $W_1 \subset W$ este subspațiu vectorial al lui W , atunci $f^{-1}(W_1)$ este subspațiu vectorial al lui V . În particular, $\text{Ker } f = \{x \in V \mid f(x) = 0\} = f^{-1}(\{0_W\})$ este subspațiu vectorial al lui V .

Propoziție. Fie $V_1, V_2 \subset V$ subspații vectoriale ale lui V . Atunci $V_1 \cap V_2$ este subspațiu vectorial al lui V .

Generalizare. Dacă V_i , $i \in I$ este subspațiu vectorial al lui V , atunci $\bigcap_{i \in I} V_i$ este subspațiu vectorial al lui V .

Exemplu. Dacă $V = \mathbb{R}^3$, V_1 este dreaptă vectorială și V_2 este planul vectorial, atunci $V_1 \cap V_2 = \{0_V\}$ sau $V_1 \cap V_2 = V_1$.

Definiție. Fie $V_1, V_2 \subset V$ subspații vectoriale ale lui V . Se numește **uniunea lui V_1 cu V_2** și se notează $V_1 + V_2$, subspațiu generat de $V_1 \cup V_2$.

Teorema dimensiunii. Fie V un spațiu vectorial peste corpul K cu $\dim_K V \neq \infty$. Fie V_1 și V_2 subspații vectoriale ale lui V . Atunci

$$\dim_K(V_1 + V_2) + \dim_K(V_1 \cap V_2) = \dim_K V_1 + \dim_K V_2.$$

Observație. Dacă $V_1 \cap V_2 = \{0_V\}$, atunci notăm $V_1 + V_2 = V_1 \oplus V_2$ și

$$\dim_K(V_1 \oplus V_2) = \dim_K V_1 + \dim_K V_2.$$

Exemplu. $W = \{(x, y) \in \mathbb{R}^2 \mid ax + by = 0, a^2 + b^2 > 0\} \subset \mathbb{R}^2$ este subspațiu vectorial peste corpul \mathbb{R}^2 .

V. SPAȚII DE MORFISME

V.1. Dualul unui spațiu vectorial

Definiție. Fie V spațiu vectorial peste corpul K . Considerăm

$$V^* = \{m : V \rightarrow K \mid m \text{ morfism de } K - \text{spatii vectoriale}\},$$

adică m este formă liniară. Pentru $m, n \in V^*$, definim $m+n : V \rightarrow K$, $(m+n)(x) = m(x) + n(x)$. De asemenea, pentru $a \in K$ și $m : V \rightarrow K$, fie $am : V \rightarrow K$, $(am)(x) = am(x)$. V^* astfel definit numește **dualul** spațiului vectorial V .

Fie V un spațiu vectorial peste corpul K , $\dim_K V = n$, $\{e_1, \dots, e_n\}$ bază a spațiului vectorial V . Atunci: există și este unică baza $\{\lambda^1, \dots, \lambda^n\} \subset V^*$, $\lambda^i : V \rightarrow K$, astfel încât $\lambda^i(e_j) = \delta_{ij}$, $\forall i, j = \overline{1, n}$. Pentru aceste funcții, dacă $x = x^j e_j$, avem

$$\lambda^i(x) = \lambda^i(x^j e_j) = x^j \lambda^i(e_j) = x^j \delta_{ij} = \begin{cases} x^j & \text{pentru } i = j \\ 0 & \text{pentru } i \neq j \end{cases}.$$

Definiție. Baza $\{\lambda^1, \dots, \lambda^n\}$ din spațiul vectorial dual V^* se numește **baza duală** bazei $\{e_1, \dots, e_n\}$ din spațiul vectorial V .

V.2. Spații de morfisme

Fie V și W spații vectoriale peste corpul K . Notăm

$$\text{Hom}_K(V, W) = \{f : V \rightarrow W \mid f \text{ morfism}\}.$$

Dacă $W = K$, atunci $\text{Hom}_K(V, K) = V^*$. Dacă $a \in K$, $f, g \in \text{Hom}_K(V, W)$ definim $(f+g)(x) = f(x) + g(x)$ și $(af)(x) = af(x)$, $\forall x \in V$.

Definiție.

1. $f \in V^*$ se numește **formă liniară** dacă $\forall a, b \in K$, $\forall x, y \in V$, $f(ax + by) = af(x) + bf(y)$.
2. $f \in \text{Hom}_K(V \times V, K) = (V \times V)^* = (V^2)^*$ se numește **formă biliniară** dacă $\forall a, b \in K$, $\forall x, y, z \in V$, $f(ax + by, z) = af(x, z) + bf(y, z)$.

3. Dacă $V^p = \underbrace{V \times \dots \times V}_{p\text{ ori}}^{not}$, în condiții similare, $f \in (V^p)^*$ se numește **formă p -liniară**.

Observație: Fie V un spațiu vectorial n -dimensional și W un spațiu vectorial m -dimensional. Dacă $\{e_1, \dots, e_n\}$ este o bază a lui V și $\{f_1, \dots, f_m\}$ o bază a lui W , atunci pentru $\varphi \in \text{Hom}_K(V, W)$ avem $\varphi(e_i) = a_i^j f_j$, $i = \overline{1, n}$,

$$\text{unde } A = \begin{pmatrix} a_1^1 & \cdots & a_1^m \\ \vdots & \ddots & \vdots \\ a_n^1 & \cdots & a_n^m \end{pmatrix} \in M_{n,m}(K)$$

Definiție. A se numește **matricea asociată morfismului** φ , relativ la bazele $\{e_1, \dots, e_n\}$ și $\{f_1, \dots, f_m\}$.

Observație. Pentru baze fixate există o bijecție între $\text{Hom}_K(V, W)$ și $M_{n,m}(K)$.

Definiție. Se numește **rangul morfismului** $\varphi \in \text{Hom}_K(V, W)$, rangul matricei asociate.

Convenție. Vom nota $\text{rang } \varphi = \text{rang } A$ într-o bază arbitrară.

Exemplu. Fie $\varphi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$, $\varphi(x^1, x^2) = (2x^1 - 3x^2, -x^1, x^1 + x^2)$.

1. φ este formă liniară
2. $A = \begin{pmatrix} 2 & -1 & 1 \\ -3 & 0 & 1 \end{pmatrix}$
3. $\text{rang } \varphi = \text{rang } A = 2$

V.3. Forme p -liniare simetrice sau alternate

Notăm $S_p = \{\sigma : \{1, 2, \dots, p\} \rightarrow \{1, 2, \dots, p\} \mid \sigma \text{ bijectivă}\}$.

Definiție. Fie V un spațiu vectorial peste corpul K . Fie $f \in (V^p)^*$.

1. f se numește **simetrică** dacă $\forall \sigma \in S_p, \forall x_1, \dots, x_p \in V$, avem $f(x_{\sigma(1)}, \dots, x_{\sigma(p)}) = f(x_1, \dots, x_p)$.
2. f se numește **alternată** dacă $\forall \sigma \in S_p, \forall x_1, \dots, x_p \in V$, avem $f(x_{\sigma(1)}, \dots, x_{\sigma(p)}) = \text{sgn}(\sigma) f(x_1, \dots, x_p)$.

Propoziție. Fie $f \in \text{Hom}_K(V, W)$. Atunci f este injectivă dacă și numai dacă $\text{Ker } f = \{0_V\}$.

Propoziție. Dacă V și W sunt spații vectoriale peste corpul K , ambele de dimensiune finită ($\dim_K V < \infty$ și $\dim_K W < \infty$) și $f \in \text{Hom}_K(V, W)$, atunci

$$\dim_K \text{Ker } f + \dim_K \text{Im } f = \dim_K V.$$

Definim: $\bar{e}_i = f(e_i) \in W$, $i = \overline{1, n}$ și $\bar{B} = \{\bar{e}_1, \dots, \bar{e}_n\} \subset \text{Im } f = f(V)$.

V.4. Forme biliniare, forme pătratice, varietăți pătratice

Fie V un spațiu vectorial peste corpul K de dimensiune n . Fie $g \in \text{Hom}_K(V^2, K)$ o formă biliniară simetrică pe V , adică $\forall x, y \in V$, $g(x, y) = g(y, x)$ și $\alpha, \beta \in K$, $\forall x, y, z \in V$ să aibă loc relația $g(\alpha x + \beta y, z) = \alpha g(x, z) + \beta g(y, z)$.

Fie $\{e_1, \dots, e_n\}$ bază a lui V și $g_{ij} = g(e_i, e_j)$ componentele lui g în baza dată. Dacă $x = (x^1, \dots, x^n) = \sum_{i=1}^n x^i e_i$ și $y = (y^1, \dots, y^n) = \sum_{j=1}^n y^j e_j$, atunci $g(x, y) = \sum_{i,j=1}^n g_{ij} x^i y^j$. Astfel asociem formei biliniare g o matrice (unică pentru o bază fixată) $G = (g_{ij})_{i,j=\overline{1,n}} \in M_{n,n}(K)$.

Definitie. Fie g o formă biliniară.

1. g se numește **pozitiv definită** dacă $g(x, x) \geq 0$, $\forall x \in V$ și dacă $g(x, x) = 0$, atunci $x = 0$.
2. g se numește **negativ definită** dacă $g(x, x) \leq 0$, $\forall x \in V$ și dacă $g(x, x) = 0$, atunci $x = 0$.
3. g se numește **semipozitiv definită** dacă $g(x, x) \geq 0$, $\forall x \in V$.
4. g se numește **seminegativ definită** dacă $g(x, x) \leq 0$, $\forall x \in V$.

Observații.

1. Dacă g este pozitiv definită, atunci $(-g)$ este negativ definită.
2. Dacă g este semipozitiv definită, atunci $(-g)$ este seminegativ definită.

Definiție. O formă biliniară g se numește **formă biliniară nedegenerată** dacă din $g(x_0, y) = 0$, $\forall y \in V$ rezultă $x_0 = 0$ (în caz contrar g este **formă biliniară degenerată**).

Observație. g este formă biliniară degenerată dacă $\exists x_0 \neq 0$ astfel încât $g(x_0, y) = 0$, $\forall y \in V$.

Fie g o formă biliniară simetrică pe K^n . Definim $q:K^n \rightarrow K$ prin $q(x) = g(x, x)$.

Definiție. Fie g o formă biliniară pe K^n . Definim $q:K^n \rightarrow K$ prin $q(x) = g(x, x)$. q se numește **formă pătratică asociată formei biliniare** g .

Observație. Dacă matricea G asociată formei biliniare are formă diagonală, spunem că forma pătratică q este scrisă în forma normală (sau canonica).

dacă $g_{i,j} = 0$ pentru $i \neq j$, atunci $q(x) = g_{11}(x^1)^2 + \dots + g_{nn}(x^n)^2$. (*)

Propoziție. $\text{rang } q = \text{rang } G$. Dacă în plus q se scrie în forma canonica (*), atunci $\text{rang } q$ este egal cu numărul scalarilor g_{11}, \dots, g_{nn} nenuli.

Observație. Printr-o eventuală renumerotare a indicilor, putem presupune că $\text{rang } q = r$ și $g_{11} \neq 0, \dots, g_{rr} \neq 0, g_{r+1,r+1} = 0, \dots, g_{nn} = 0$.

Lema lui Gauss. Fie V un spațiu vectorial peste corpul \mathbb{R} , $\dim_{\mathbb{R}} V = n$ și fie q o formă pătratică asociată unei forme biliniare simetrice g . Atunci există o bază a lui V astfel încât q să se scrie în forma canonica (*).

Teorema lui Sylvester (teorema indicelui). Fie q o formă pătratică în \mathbb{R}^n și $\text{rang } q = r$. Atunci indicele negativ de inerție este independent de baza aleasă.

Exemplu. Să se aducă la forma canonica aplicația $q: \mathbb{R}^2 \rightarrow \mathbb{R}$, $q(x_1, x_2) = (x_1)^2 - x_1 x_2 - (x_2)^2$ (folosind metoda lui Gauss). Avem:

$$q(x_1, x_2) = \left(x_1 - \frac{x_2}{2} \right)^2 - \frac{x_2^2}{4} - x_2^2 = \left(x_1 - \frac{1}{2} x_2 \right)^2 - \frac{5}{4} (x_2)^2.$$

Notând $z_1 = x_1 - \frac{1}{2} x_2$ și $z_2 = \frac{\sqrt{5}}{2} x_2$, rezultă că avem $q(z) = (z_1)^2 - (z_2)^2$.

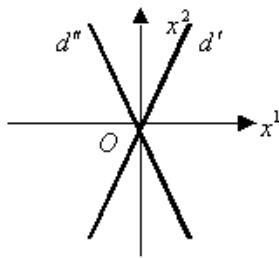
V.5. Clasificarea varietăților pătratice în \mathbb{R}

Definiție. Fie q o formă pătratică pe \mathbb{R}^n . Se numește **varietate pătratică** asociată lui q mulțimea $\{x \in \mathbb{R}^n \mid q(x) = 0\}$.

Exemplu. În \mathbb{R}^2 considerăm $f(x, y) = x^1 y^1 - 2x^2 y^2$, iar forma pătratică asociată este $q(x) = (x^1)^2 - 2(x^2)^2$. Varietatea pătratică asociată este mulțimea $\{x \in \mathbb{R}^2 \mid (x^1)^2 - 2(x^2)^2 = 0\}$. Din $(x^1)^2 - 2(x^2)^2 = 0$ avem

$(x^1 - \sqrt{2}x^2)(x^1 + \sqrt{2}x^2) = 0$ ($x^1 = \sqrt{2}$, adică obținem ecuațiile a două drepte
 $d': x^1 - \sqrt{2}x^2 = 0$ și $d'': x^1 + \sqrt{2}x^2 = 0$.

Grafic avem următoarea situație:



Observație. Fie q o formă pătratică din \mathbb{R}^n , rang $q = r$ și indice negativ de inerție p . Într-o bază (judicios aleasă) din \mathbb{R}^n avem

$$q(x) = -(x^1)^2 - \dots - (x^p)^2 + (x^{p+1})^2 + \dots + (x^r)^2.$$

Din $q(x) = 0$, obținem

$$-(x^1)^2 - \dots - (x^p)^2 + (x^{p+1})^2 + \dots + (x^r)^2 = 0. \quad (1)$$

Varietățile pătratice de forma $q(x) = 0$ se studiază prin soluțiile ecuației (1) (modulo o schimbare de coordonate).

V.5.1. Clasificarea varietăților pătratice pentru $n=2$: \mathbb{R}^2

Fie

$$f(x, y) = f_{11}x^1y^1 + f_{12}x^1x^2 + f_{21}x^2y^1 + f_{22}x^2y^2,$$

iar

$$q(x) = f_{11}(x^1)^2 + (f_{12} + f_{21})x^1x^2 + f_{22}(x^2)^2,$$

ceea ce pentru formele biliniare simetrice ($f_{12} = f_{21}$) revine la

$$q(x) = f_{11}(x^1)^2 + 2f_{12}x^1x^2 + f_{22}(x^2)^2.$$

Formele pătratice sunt:

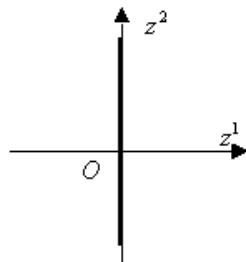
- 1) $q(z) = (z^1)^2 + (z^2)^2$ pentru $r = 2$, $p = 0$;
- 2) $q(z) = -(z^1)^2 + (z^2)^2$ pentru $r = 2$, $p = 1$;
- 3) $q(z) = -(z^1)^2 - (z^2)^2$ pentru $r = 2$, $p = 2$;
- 4) $q(z) = (z^1)^2$ pentru $r = 1$, $p = 0$;
- 5) $q(z) = -(z^1)^2$ pentru $r = 1$, $p = 1$;
- 6) $q(z) = 0$ pentru $r = 0$.

Varitățile pătratice asociate sunt:

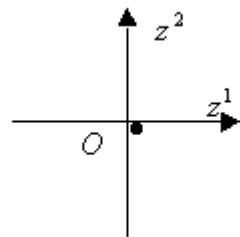
- a) $\{z \in \mathbb{R}^2 \mid q(z) = 0\} = \mathbb{R}^2$ (pentru 6)
- b) $\{z \in \mathbb{R}^2 \mid (z^1)^2 = 0\}$ = varietatea pătratică este formată din două drepte confundate (pentru 5 și 5)
- c) $\{z \in \mathbb{R}^2 \mid (z^1)^2 + (z^2)^2 = 0\}$ = varietatea pătratică este formată din două puncte confundate (pentru 1 și 3)
- d) $\{z \in \mathbb{R}^2 \mid (z^1)^2 - (z^2)^2 = 0\}$ = varietatea pătratică este formată din două drepte (pentru 2)

Reprezentările grafice pentru aceste cazuri sunt:

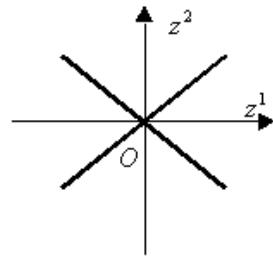
b)



c)



d)



V.5.2. Clasificarea varietăților pătratice pentru $n=3$: \mathbb{R}^3

Fie forma pătratică simetrică:

$$q(x) = f_{11}(x^1)^2 + 2f_{12}x^1x^2 + 2f_{13}x^1x^3 + f_{22}(x^2)^2 + 2f_{23}x^2x^3 + f_{33}(x^3)^2.$$

În forma normală avem

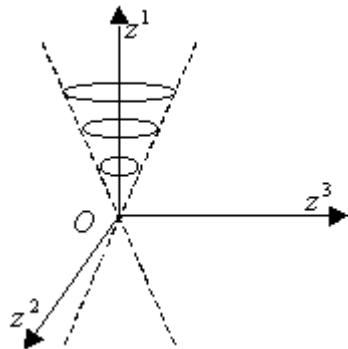
$$q(z) = \lambda_1(z^1)^2 + \lambda_2(z^2)^2 + \lambda_3(z^3)^2,$$

unde $\lambda_1, \lambda_2, \lambda_3 \in \{-1, 0, 1\}$.

Formele și varietățile pătratice sunt:

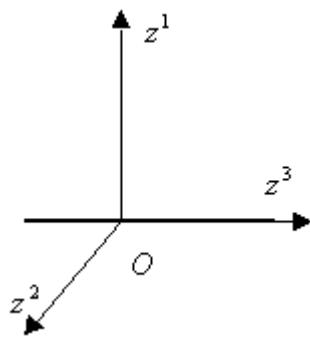
1. $(z^1)^2 + (z^2)^2 + (z^3)^2 = 0$ pentru $r = 3$, $p = 0$; varietatea pătratică este $\{(0,0,0)\}$, punct dublu;
2. $-(z^1)^2 + (z^2)^2 + (z^3)^2 = 0$ pentru $r = 3$, $p = 1$; varietatea pătratică este conul de ecuație $(z^1)^2 = (z^2)^2 + (z^3)^2$ cu axa Oz^1 ;

Reprezentarea grafică este:

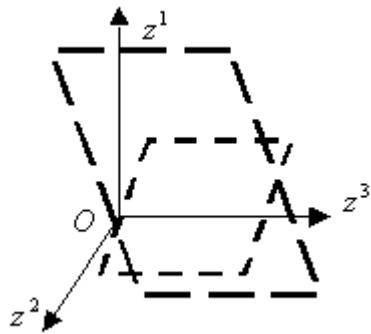


3. $-(z^1)^2 - (z^2)^2 + (z^3)^2 = 0$ pentru $r = 3$, $p = 2$; varietatea pătratică este conul de ecuație $(z^3)^2 = (z^1)^2 + (z^2)^2$ cu axa Oz^3 ;
4. $-(z^1)^2 - (z^2)^2 - (z^3)^2 = 0$ pentru $r = 3$, $p = 3$; varietatea pătratică este $\{(0,0,0)\}$, punct dublu;
5. $(z^1)^2 + (z^2)^2 = 0$ pentru $r = 2$, $p = 0$; varietatea pătratică este $\{z \in \mathbb{R}^3 \mid z^1 = z^2 = 0\}$, axa Oz^3 dublă;

Reprezentarea grafică este:



6. $-(z^1)^2 + (z^2)^2 = 0$ pentru $r = 2$, $p = 1$; varietatea pătratică este formată din două planuri vectoriale care conțin Oz^3 , planul π , de ecuație $z^1 = z^2$ și planul $\bar{\pi}$, de ecuație $z^1 = -z^2$;



7. $-(z^1)^2 - (z^2)^2 = 0$ pentru $r = 2$, $p = 2$; varietatea pătratică este aceeași ca la 5.
8. $(z^1)^2 = 0$ pentru $r = 1$, $p = 0$; varietatea pătrativă este planul dublu de ecuație $z^1 = 0$ (planul z^2Oz^3);
9. $-(z^1)^2 = 0$ pentru $r = 1$, $p = 1$; varietatea pătratică este similară cu cea de la 8;
10. $q(z) = 0$ pentru $r = 0$; varietatea pătratică este $\{z \in \mathbb{R}^3 \mid q(z) = 0\} = \mathbb{R}^3$.

VI. SPĂȚII AFINE

Fie K un corp comutativ. Fie V un spațiu vectorial peste corpul K și A o mulțime nevidă. Fie $\varphi: A \times A \rightarrow V$ o funcție, care se numește **funcție de structură**.

Definiție. (A, V, φ) se numește **spațiu afin de spațiu director V și funcție de structură φ** dacă:

- 1) $\varphi(P, Q) + \varphi(Q, S) = \varphi(P, S)$, $\forall P, Q, S \in A$ (relația lui Chasles);
- 2) $\forall P \in A$, $\varphi(P, \bullet) \stackrel{\text{not}}{=} \varphi_P : A \rightarrow V$ este bijectivă.

Convenție. Elementele $v, w, \dots \in V$ se numesc **vectori**, $P, Q, S, \dots \in A$ se numesc **puncte**, $a, b, \dots \in K$ se numesc **scalari**.

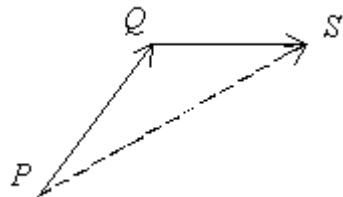
Notăție. $\varphi(P, Q) \stackrel{\text{not}}{=} PQ = \overline{PQ} = \vec{PQ}$.

$(P, Q) \in A \times A$ se numește **bipunct**.



Observații.

1. Relația lui Chasles se rescrie $\overline{PQ} + \overline{QS} = \overline{PS}$.



2. Fie (A, V, φ) spațiu afin (implicit V , K și φ vor fi subînțelese).

VI.1. Centre de greutate (baricentre)

Propoziție. Fie A spațiu afin ca mai sus; $S = \{A_0, A_1, \dots, A_p\} \subset A$, $\alpha^0, \alpha^1, \dots, \alpha^p \in K$ astfel încât $\alpha^0 + \alpha^1 + \dots + \alpha^p = 1$. Atunci există $P \in A$ astfel încât pentru $O \in A$ să avem

$$OP = \alpha^0 OA_0 + \dots + \alpha^p OA_p \quad (*)$$

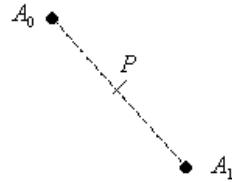
P se numește **centrul de greutate (baricentrul)** sistemului S , cu ponderile $\alpha^0, \alpha^1, \dots, \alpha^p$.

Notăție.

$$P = \alpha^0 A_0 + \alpha^1 A_1 + \dots + \alpha^p A_p \quad (**)$$

Exemplu. Fie $A = \mathbb{R}^3$, $V = \mathbb{R}^3$, $K = \mathbb{R}$ și $S = \{A_0, A_1\}$. Pentru

$$\alpha^0 = \alpha^1 = \frac{1}{2}$$
 se obține mijlocul segmentului $\overline{A_0 A_1}$, $P = \frac{1}{2} A_0 + \frac{1}{2} A_1$.



VI.2. Punkte afin dependente (independente)

Definiție. Fie A un spațiu afin și $S = \{A_0, A_1, \dots, A_p\} \subset A$. S se numește **sistem de puncte afin dependente** dacă există ponderile $\alpha^1, \dots, \alpha^p$ din K cu $\alpha^1 + \dots + \alpha^p = 1$ astfel încât $A_0 = \alpha^1 A_1 + \dots + \alpha^p A_p$ (alegerea lui A_0 este arbitrară).

În caz contrar, S se numește **sistem de puncte afin independente**.

Propoziție. Fie A spațiu afin și $S = \{A_0, A_1, \dots, A_p\} \subset A$. Atunci S este afin dependent dacă și numai dacă $S = sp\{A_0 A_1, \dots, A_0 A_p\}$ este liniar dependent.

VI.3. Subspații affine

Definiție. Fie A un spațiu afin cu spațiul vectorial director V și funcție de structură $\varphi : A \times A \rightarrow V$. Fie $A' \subset A$ și $\varphi' = \varphi|_{A' \times A'} : A' \times A' \rightarrow V$. A' se numește **subspațiu afin** al lui A dacă $\varphi(A' \times A') = V'$ este subspațiu vectorial al lui V , adică (A', V', φ') este spațiu afin cu spațiul vectorial director V' și funcție de structură φ' .

Observație. Fie (A, V, φ) spațiu afin și A' submulțime în A . Atunci A' este subspațiu afin dacă și numai dacă $\forall \alpha, \beta \in K$, astfel încât $\alpha + \beta = 1$ și $\forall P', Q' \in A'$ avem $\alpha P' + \beta Q' \in A'$.

Propoziție. Fie (A, V, φ) un spațiu afin, A', A'' subspații affine ale lui A . Atunci $A' \cap A''$ este subspațiu afin al lui A . Dacă $A' \cap A'' \neq \emptyset$, atunci spațiul său director este $V' \cap V''$, unde $V' = \varphi(A' \times A')$ și $V'' = \varphi(A'' \times A'')$.

VII. REPERE CARTEZIENE. REPERE AFINE. DREPTE. HIPERPLANE

VII.1. Repere carteziene și affine

Fie (A, V, φ) un spațiu afin de spațiu director V și funcție de structură φ . V este un spațiu vectorial peste corpul K și $\dim A = \dim_K V = n$. φ este funcția de structură astfel încât: $\varphi(P, Q) = PQ = \overline{PQ} = \vec{PQ}$. Fie $P_0 \in A$; $P_0 P \in V$. Fixăm o bază $\{e_1, \dots, e_n\}$ a lui V . Atunci există $p^1, \dots, p^n \in K$ astfel încât $P_0 P = p^1 e_1 + \dots + p^n e_n = (p^1, \dots, p^n)$.

Definiție. Se numește **reper cartezian** al lui A de origini (centru) P_0 și de direcții e_1, \dots, e_n mulțimea $R = \{P_0; e_1, \dots, e_n\}$.

Definiție. Fie $P_0, P_1, \dots, P_n \in A$ puncte affine independente (adică $P_0 P_1, \dots, P_0 P_n$ sunt liniar independenți în V) $\{P_0; P_1, \dots, P_n\}$ dată în observația de mai sus se numește **reper afin determinat de punctele affine independente** P_0, \dots, P_n dacă pentru orice $P \in A$, există x^1, \dots, x^n astfel încât să avem $\overline{P_0 P} = x^1 \overline{P_0 P_1} + \dots + x^n \overline{P_0 P_n}$.

Observație. Fie $P_0, P_1, \dots, P_n \in A$ puncte affine independente, atunci $R = \{P_0; P_0 P_1, \dots, P_0 P_n\}$ este reper cartezian de centru P_0 și bază $e_1 = \overline{P_0 P_1}$.

VII.2. Drepte în spațiul afin K^n

Fie P_0 și reperul cartezian $\{P_0; e_1, \dots, e_n\}$, unde $\{e_1, \dots, e_n\}$ este bază în K^n . Atunci $(P_0 P_0) = (0, \dots, 0)$. Presupunem (pentru simplitate) că $P_0 = (0, \dots, 0)$.

VII.2.1. Ecuația dreptei ce trece prin două puncte distincte

Fie două puncte $P_1(x_0^1, \dots, x_0^n)$ și $P_2(y_0^1, \dots, y_0^n)$. Avem

$$\begin{aligned} \varphi(P_1, P_2) &= P_2 - P_1 = (y_0^1 - x_0^1, \dots, y_0^n - x_0^n) = \overline{P_1 P_2}; \\ \overline{P_1 P_0} + \overline{P_0 P_1} &= \overline{P_1 P_1}; \quad \overline{P_1 P_2} = \overline{P_1 P_0} + \overline{P_0 P_2} = \overline{P_0 P_2} - \overline{P_0 P_1}. \end{aligned}$$

Cum $P_1 \neq P_2$, atunci există $i_0 \in \{1, 2, \dots, n\}$ astfel încât $y_0^{i_0} - x_0^{i_0} \neq 0$ (deci există o soluție nenulă).

De asemenea, considerăm $V' = sp(P_1, P_2) \subset K^n$, care este 1-dimensional (adică este dreapta vectorială $V' = \{\lambda \overline{P_1 P_2} | \lambda \in K\}$). Atunci, $P \in A'$ dacă și numai dacă $P_0 P \in V'$.

Dacă $P(z^1, \dots, z^n)$, avem $z^i = (y_0^i - x_0^i)$. Luăm $P = P_1$, $P_1 \in A_1$. Pentru $P \in A_1$

$$P - P_1 = \lambda \overline{P_1 P_2} = \lambda (y_0^1 - x_0^1, \dots, y_0^n - x_0^n)$$

$$P_0 - P_1 = (x_0^1 - z^1, \dots, x_0^n - z^n), z^i - x_0^i = \lambda (y_0^i - x_0^i), i = \overline{1, n}$$

astfel că

$$(-\lambda) = \frac{z^i - x_0^i}{y_0^i - x_0^i}.$$

Aveam

$$\frac{z^1 - x_0^1}{y_0^1 - x_0^1} = \frac{z^2 - x_0^2}{y_0^2 - x_0^2} = \dots = \frac{z^n - x_0^n}{y_0^n - x_0^n} (= -\lambda). \quad (1)$$

Convenție. Dacă unul dintre numitorii este zero, prin convenție presupunem și că numărătorul respectiv este tot zero.

VII.2.2. Ecuația dreptei care trece printr-un punct și are direcția dată

Fie (z^1, \dots, z^n) coordonate în K^n , $v = (v_0^1, \dots, v_0^n)$ direcția (nenulă) dată și (x_0^1, \dots, x_0^n) coordonatele punctului fixat. Atunci

$$\frac{z^1 - x_0^1}{v_0^1} = \frac{z^2 - x_0^2}{v_0^2} = \dots = \frac{z^n - x_0^n}{v_0^n} \quad (2)$$

(unde $v_0^i = y_0^i - x_0^i$).

Fie dreapta d de ecuație $\frac{z^1 - x_0^1}{v^1} = \dots = \frac{z^n - x_0^n}{v^n}$ și $P_0(x_0^1, \dots, x_0^n) \in d$.

Spațiul director al lui d este $\{\lambda v | \lambda \in K\}$, unde $v = (v^1, \dots, v^n)$. De asemenea, considerăm și dreapta \bar{d} de ecuație: $\frac{z^1 - \bar{x}_0^1}{\bar{v}^1} = \dots = \frac{z^n - \bar{x}_0^n}{\bar{v}^n}$, unde $\bar{P}_0(\bar{x}_0^1, \dots, \bar{x}_0^n) \in \bar{d}$, iar spațiul director al lui \bar{d} este $\{\lambda \bar{v} | \lambda \in K\}$ cu $\bar{v} = (\bar{v}^1, \dots, \bar{v}^n)$.

În aceste condiții, $d \parallel \bar{d}$ dacă și numai dacă v și \bar{v} sunt coliniari (**Două drepte sunt paralele dacă au aceeași parametri directori**).

Observație. Dacă $P_0 \in d$, direcția $v \neq 0$ și $\frac{z^1 - x_0^1}{v^1} = \dots = \frac{z^n - x_0^n}{v^n} = t$,

atunci $\begin{cases} z^1 - x_0^1 = tv^1 \\ \vdots \\ z^n - x_0^n = tv^n \end{cases}$, adică $\begin{cases} z^1 = x_0^1 + tv^1 \\ \vdots \\ z^n = x_0^n + tv^n \end{cases}$. Aceasta este **ecuația parametrică** a dreptei.

Observație. Dacă $n = 2$, atunci ecuația dreptei este $\frac{z^1 - x_0^1}{v^1} = \frac{z^2 - x_0^2}{v^2}$,

$$v^1 \neq 0, \text{ adică } z^2 = \frac{v^2}{v^1}(z^1 - x_0^1) + x_0^2 \text{ (ecuația explicită a dreptei în } \mathbb{R}^2\text{).}$$

VII.3. Hiperplan afin în \mathbb{R}^n

Fie H subspațiu afin de dimensiune $n-1$. Notăm H cu A' . $V^1 \subset V$ hiperspațiu vectorial corespunzător din \mathbb{R}^n .

Observații.

- 1) Dacă $n = 2$, atunci hiperplanul este o dreaptă.
- 2) Dacă $n = 3$, atunci hiperplanul este un plan.

Fie $B^1 = \{v_1, \dots, v_{n-1}\}$ bază în V^1 , $\begin{cases} v_1 = \{v_1^1, \dots, v_1^n\} \\ \vdots \\ v_{n-1} = \{v_{n-1}^1, \dots, v_{n-1}^n\} \end{cases}$. Atunci A' se

scrie $A' : \begin{vmatrix} z^1 - x_0^1 & z^2 - x_0^2 & \dots & z^n - x_0^n \\ v_1^1 & v_1^2 & \dots & v_1^n \\ \vdots & \vdots & \ddots & \vdots \\ v_{n-1}^1 & v_{n-1}^2 & \dots & v_{n-1}^n \end{vmatrix} = 0$. În particular, pentru $n = 2$

avem $\begin{vmatrix} z^1 - x_0^1 & z^2 - x_0^2 \\ v_1^1 & v_1^2 \end{vmatrix} = 0$ dacă și numai dacă $v_1^2(z^1 - x_0^1) + v_1^1(z^2 - x_0^2) = 0$

sau $v_1^2 z^1 + v_1^1 z^2 + a = 0$ (ecuație implicită). Pentru $n = 3$ avem

$\begin{vmatrix} z^1 - x_0^1 & z^2 - x_0^2 & z - x_0^3 \\ v_1^1 & v_1^2 & v_1^3 \\ v_2^1 & v_2^2 & v_2^3 \end{vmatrix} = 0$. De aici obținem $a_1 z^1 + a_2 z^2 + a_3 z^3 + b = 0$,

ecuația canonica a unui plan în \mathbb{R}^3 .

În general, $a_1 z^1 + \dots + a_n z^n + b = 0$ (unde nu toți a_1, \dots, a_n sunt zero).

VIII. FORME BIAFINE. FORME PATRATICE AFINE. HIPERCUADRICE ÎN SPAȚII AFINE

Definiție. $g : V \rightarrow K$ este formă pătratică asociată lui φ , dacă

$$K^n$$

$$g(x) = \varphi(x, x) = \sum_{i,j=1}^n a_{ij} x^i x^j.$$

Teorema Gauss-Lagrange. Există o bază a lui K^n în care g să aibă formă canonică (normală) $g(z) = \lambda_1(z^1)^2 + \dots + \lambda_n(z^n)^2$, $\lambda_i \in K$, $i = \overline{1, n}$. Dacă $\text{ang } g = \text{rang } \varphi = \text{rang}[(a_{ij})_{i,j=\overline{1,n}}] = r$, atunci putem alege $\lambda_1 \neq 0$, ..., $\lambda_r \neq 0$, $\lambda_{r+1} = \dots = \lambda_n = 0$.

Observații.

1. Dacă $K = \mathbb{C}$ putem reduce pe g la forma $g(z) = (z^1)^2 + \dots + (z^r)^2$.
2. Dacă $K = \mathbb{R}$, $g(z) = (z^1)^2 + \dots + (z^p)^2 - (z^{p+1})^2 - \dots - (z^r)^2$, $p \leq r \leq n$, unde p este indicele pozitiv de inerție.

Definiție. Se numește **formă biafină** pe \mathbb{R}^n o aplicație $F : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ afină în fiecare argument, $F(x, y) = \sum a_{ij} x^i y^j + \sum b_i x^i + \sum c_i y^i$, $x, y \in \mathbb{R}^n$.

Vom lucra numai cu forme biafine simetrice, adică $F(x, y) = F(y, x)$.

Observație. Pentru formele biafine simetrice avem:

1) $a_{ij} = a_{ji}$ și $b_i = c_i$, $\forall i, j = \overline{1, n}$.

2) $F : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$,

$$F(x, y) = \sum a_{ij} x^i x^j + \sum b_i x^i + \sum c_i y^i + c.$$

Forma pătratică afină asociată lui F este $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $f(x) = F(x, x)$ și deoarece $a_{ij} = a_{ji}$ avem $f(x) = 2 \sum a_{ij} x^i x^j + 2 \sum b_i x^i + c$

Notăm $A = (a_{ij})_{i,j=\overline{1,n}} \in M_{n,n}(R)$, $A = {}^t A$; $b = (b_1, \dots, b_n)$,

$$\begin{pmatrix} & & b_1 \\ A & & \vdots \\ & & b_n \\ b_1 & \cdots & b_n & -c \end{pmatrix} \in M_{(n+1),(n+1)}(R)$$

și avem $r = \text{rang } A = \text{rang } \tilde{f} = \text{rang } \tilde{F}$, $\rho = \text{rang } D = \text{rang } F = \text{rang } F$.

Notăm $\tilde{F}(x, y) = \sum a_{ij} x^i y^j$, $\tilde{f}(x) = \sum a_{ij} x^i x^j$.

Definiție. Fie $f: \mathbb{R}^n \rightarrow \mathbb{R}$ o formă pătratică afină $f(x) = \sum a_{ij}x^i x^j + 2\sum b_i x^i + c$, unde $a_{ij} = a_{ji}$. Se numește **hipercuadrică asociată lui f** , $F = \{x \in \mathbb{R}^n \mid f(x) = 0\}$.

Hipercuadricele în \mathbb{R}^n se clasifică după r , p și ρ . Dacă $n = 2$, hipercuadricele se numesc **conice**. Dacă $n = 3$, hipercuadricele se numesc **cuadrice**.

VIII.1. Clasificarea afină a conicelor $n = 2$ (\mathbb{R}^2)

Clasificarea conicelor din \mathbb{R}^2 este dată în următorul tabel:

ρ	r	p	Ecuația normală	Denumirea
3	2	2	$(x^1)^2 + (x^2)^2 = 1$	elipsa
3	2	1	$(x^1)^2 - (x^2)^2 = 1$	hiperbola
3	2	0	$-(x^1)^2 - (x^2)^2 = 1$	elipsa imagină
3	1	1	$(x^1)^2 - 2x^2 = 0$	parabolă
2	2	2	$(x^1)^2 + (x^2)^2 = 0$	un punct dublu $(0,0)$
2	2	1	$(x^1)^2 - (x^2)^2 = 0$	două drepte concurente
2	1	1	$(x^1)^2 = 1$	Două drepte paralele
2	1	0	$-(x^1)^2 = 0$	Două drepte paralele imaginare
1	1	1	$(x^1)^2 = 0$	O dreaptă dublă

Exemplu. În \mathbb{R}^2 fie ecuația $(x^1)^2 + (4x^2)^2 - 2x^1 + 8x^2 - 5 = 0$.

Observăm că $D = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 4 & 4 \\ -1 & 4 & -5 \end{pmatrix}$, $r = \text{rang } A = 2$, $\rho = 3$, $\det D = -40$.

Astfel avem

$$((x^1)^2 - 2x^1 + 1) + 4((x^2)^2 + 2x^2 + 1) - 5 - 1 - 4$$

deci $(x^1 - 1)^2 + 4(x^2 + 1)^2 = 10$ și facem substituțiile $y^1 = x^1 - 1$, $y^2 = 2(x^2 + 1)$ și obținem $(y^1)^2 + (y^2)^2 = 10$, astfel că $p = 2$.

VIII.2. (Hiper)cuadrice în spațiul afin \mathbb{R}^3

Dacă $n = 3$, atunci H se numește cuadratică. În acest caz

$$D = \begin{pmatrix} & b_1 \\ A & \begin{matrix} b_2 \\ b_3 \end{matrix} \\ b_1 & b_2 & b_3 & -c \end{pmatrix}, r = \text{rang } A, \rho = \text{rang } D, p$$

este indicele pozitiv de

inertie al formei pătratice vectoriale $q(y) = \sum a_{ij}y^i y^j$. Hipercuadricele se clasifică în funcție de ρ, r și p .

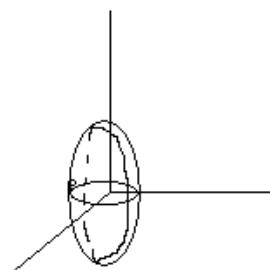
Observație. $\rho \leq n+1, p \leq r \leq n, r \leq \rho \leq r+2$.

Clasificarea cuadricelor din \mathbb{R}^3 este dată în următorul tabel:

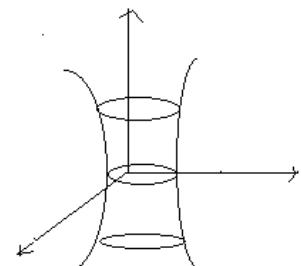
ρ	r	p	Ecuația (în formă canonica)	Denumirea
4	3	3	$(x^1)^2 + (x^2)^2 + (x^3)^2 = 1$	Elipsoid
4	3	2	$(x^1)^2 + (x^2)^2 - (x^3)^2 = 1$	Hiperboloid cu o pânză
4	3	1	$(x^1)^2 - (x^2)^2 - (x^3)^2 = 1$	Hiperboloid cu două pânze
4	3	0	$-(x^1)^2 - (x^2)^2 - (x^3)^2 = 1$	Elipsoid imaginar
3	2	2	$(x^1)^2 + (x^2)^2 = 1$	Cilindru eliptic
3	2	1	$(x^1)^2 - (x^2)^2 = 1$	Cilindru hiperbolic
3	2	0	$-(x^1)^2 - (x^2)^2 = 1$	Cilindru imaginari
4	2	2	$(x^1)^2 + (x^2)^2 - 2x^3 = 0$	Paraboloid eliptic
4	2	1	$(x^1)^2 - (x^2)^2 - 2x^3 = 0$	Paraboloid hiperbolic
3	3	3	$(x^1)^2 + (x^2)^2 + (x^3)^2 = 0$	Punct dublu
3	3	2	$(x^1)^2 + (x^2)^2 - (x^3)^2 = 0$	con
3	1	1	$(x^1)^2 - 2x^2 = 0$	Cilindru parabolic
2	2	2	$(x^1)^2 + (x^2)^2 = 0$	O dreapta dubla
2	2	1	$(x^1)^2 - (x^2)^2 = 0$	2 plane concurente
2	1	1	$(x^1)^2 = 1$	2 plane paralele
2	1	0	$-(x^1)^2 = 1$	2 plane imaginare

Reprezentările grafice sunt după cum urmează:

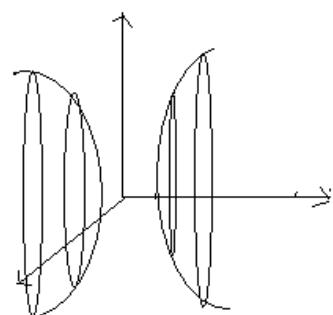
Elipsoid



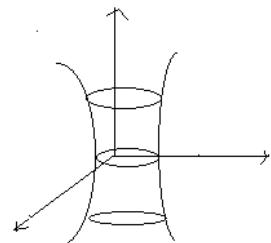
Hiperboloid
cu o pânză



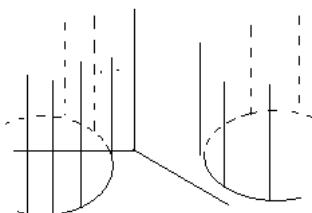
Hiperboloid
cu două
pânze

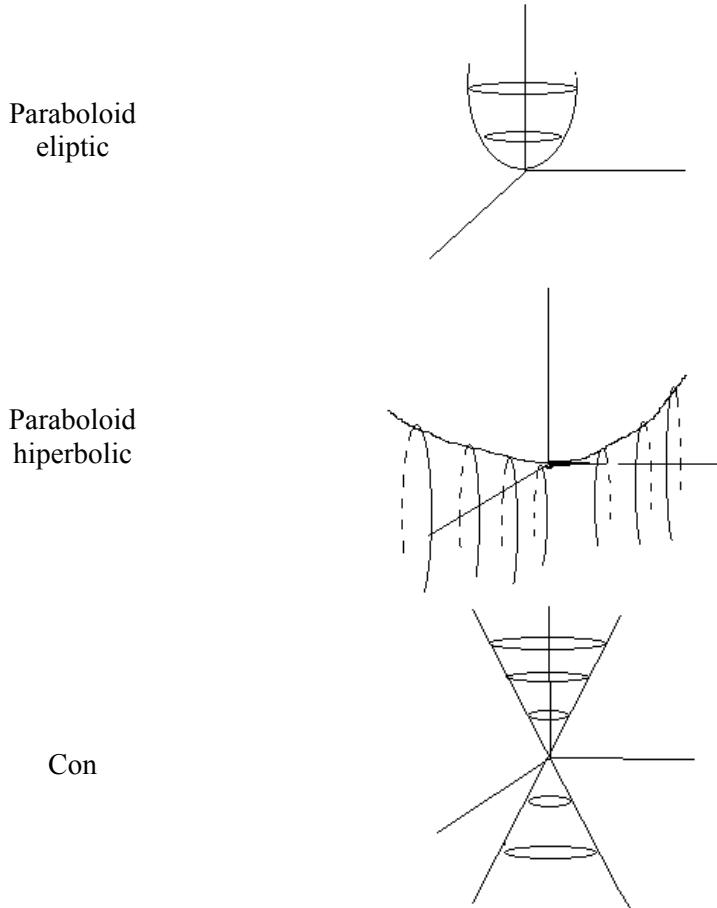


Cilindru
eliptic



Cilindru
hiperbolic





IX. SPAȚII VECTORIALE EUCLIDIENE

Fie $K = \mathbb{R}$, V un spațiu vectorial real și $g \in \text{Hom}(V \times V, \mathbb{R})$ formă biliniară pe V .

Definiție. g se numește **produs scalar (metrică)** pe V dacă g este simetrică și pozitiv definită (adică oricare ar fi $x, y \in V$, $g(x, y) = g(y, x)$; oricare ar fi $x \in V$, $g(x, x) \geq 0$ și $g(x, x) = 0 \Leftrightarrow x = 0$).

Definiție. (V, g) se numește **spațiu vectorial euclidian**.

Exemplu.

- 1) $V = \mathbb{R}^2$, $g: \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}$, $g(x, y) = x^1 y^1 + x^2 y^2$, unde $x = (x^1, x^2)$, $y = (y^1, y^2)$. Fie $x \in \mathbb{R}^2$, $g(x, x) = (x^1)^2 + (x^2)^2$. $g(x, x) = 0 \Leftrightarrow x = 0$, prin urmare (\mathbb{R}^2, g) este spațiu vectorial euclidian bidimensional.

2) Fie $V = \mathbb{R}^n$, $g : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, $g(x, y) = \sum x^i y^j$. $(\mathbb{R}^n g)$ se numește **spațiu vectorial euclidian n -dimensional canonic**.

Propoziție. Fie V, W spații vectoriale (reale) și $f : V \rightarrow W$ un morfism injectiv de spații vectoriale. Fie $g : W \times W \rightarrow \mathbb{R}$ un produs scalar pe W . Atunci $h : V \times V \rightarrow \mathbb{R}$, $h(x, y) = g(f(x), f(y))$ este produs scalar pe V .

Corolar. Fie (W, g) un spațiu vectorial euclidian și fie $V \subset W$ subspațiu vectorial. Atunci $(V, g|_{V \times V})$ este spațiu vectorial euclidian.

Exemplu. $W = \mathbb{R}^3$, $g(x, y) = x^1 y^1 + x^2 y^2 + x^3 y^3$ și $V = \mathbb{R}^2 \times \{0\}$.

Atunci $g|_{V \times V}(x, y) = x^1 y^1 + x^2 y^2$ (deci produsul scalar din \mathbb{R}^2).

Definiție. Fie (V, g) un spațiu vectorial euclidian și $v \in V$. Se numește **lungimea lui v** (sau **norma lui v**), $\|v\| = \sqrt{g(v, v)}$.

Observații.

1. $\|v\| \geq 0$;
2. radicalul este bine definit;
3. $\|v\| = 0 \Leftrightarrow v = 0$.

Definiție. Fie $v \in V$. Spunem că v este **unitar**, dacă $\|v\| = 1$.

Observație. Fie $v \neq 0$ și $\|v\| \neq 0$. Definim $w = \frac{v}{\|v\|} = \frac{1}{\|v\|} \cdot v$ și atunci

$$g(w, w) = d\left(\frac{1}{\|v\|} \cdot v, \frac{1}{\|v\|} \cdot v\right) = \frac{1}{\|v\|^2} g(v, v) = 1.$$

Propoziție (Inegalitatea lui Cauchy-Buniakowski). Fie $v, w \in V$ (spațiu vectorial euclidian), și $\langle \cdot, \cdot \rangle$ produsul scalar. Atunci

$$|\langle v, w \rangle| \leq \|v\| \cdot \|w\| \quad (1)$$

și avem egalitate dacă și numai dacă v și w sunt coliniare.

Corolar (Inegalitatea triunghiului, Inegalitatea lui Minkowski). Fie $v, w \in V$. Atunci

$$\|v + w\| \leq \|v\| + \|w\|, \quad (2)$$

cu egalitate dacă și numai dacă v și w sunt coliniari.

Corolar. Fie $x, y \in \mathbb{R}^n$. Atunci există și este unic $\theta \in [0, \pi]$ astfel încât $\cos \theta = \frac{\langle x, y \rangle}{\|x\| \cdot \|y\|}$.

X. ORTOGONALITATE

Definiție. Spunem ca x este ortogonal pe y și scriem $x \perp y$ dacă și numai dacă $\langle x, y \rangle = 0$.

Exemple.

- 1) $0 \perp x$, $\forall x \in V$;
- 2) dacă $x \perp y$, $\forall y \in V$, atunci $x = 0$;
- 3) Fie $V = \mathbb{R}^3$ și $\langle \cdot, \cdot \rangle$ produsul scalar canonic $\langle x, y \rangle = x^1y^1 + x^2y^2 + x^3y^3$.

Aveam $e_1 = (1, 0, 0)$ cu $\|e_1\|^2 = \langle e_1, e_1 \rangle = 1$, $e_2 = (0, 1, 0)$ cu $\|e_2\|^2 = \langle e_2, e_2 \rangle = 1$ și $e_3 = (0, 0, 1)$ cu $\|e_3\|^2 = \langle e_3, e_3 \rangle = 1$. În plus $\langle e_1, e_2 \rangle = \langle e_1, e_3 \rangle = \langle e_2, e_3 \rangle = 0$, de unde rezultă $e_1 \perp e_2$, $e_1 \perp e_3$ și $e_2 \perp e_3$. Suplimentar avem că $\{e_1, e_2, e_3\}$ este bază a lui $V = \mathbb{R}^3$.

Definiție. O bază $\{e_1, \dots, e_n\}$ se numește **ortonormală** dacă $\langle e_i, e_j \rangle = \delta_{ij}$,

$$\forall i, j = \overline{1, n} \text{ cu } \delta_{ij} = \begin{cases} 1 & \text{pentru } i = j \\ 0 & \text{pentru } i \neq j \end{cases}.$$

Observații.

1. Fie $x_0 \in V$ fixat, $W = \{y \in V \mid \langle y, x_0 \rangle = 0\}$ spațiu vectorial. Acest spațiu vectorial se numește **ortogonalul** lui x_0 , notat x_0^\perp .
2. Fie $W \subset V$ subspațiu în V . Atunci $W^\perp = \{y \in V \mid \langle y, w \rangle = 0, \forall w \in W\}$. Aceasta este tot un subspațiu vectorial în V . În particular, dacă $W = \mathbb{R}x_0 = \{\lambda x_0 \mid \lambda \in \mathbb{R}\}$, atunci $x_0^\perp = W^\perp$.

Procedeul de ortonormalizare Gram-Schmidt

Fie $S = \{v_1, \dots, v_n\} \subset V$, sistem de vectori liniari independenți. Fie $\langle \cdot, \cdot \rangle$ produsul scalar pe V . Formăm $e_1 = \frac{1}{\|v_1\|} \cdot v_1$ unitar și $sp\{v_1\} = sp\{e_1\}$ și $sp\{v_1\}$, $sp\{e_1\}$ sunt subspații la fel orientate. Luăm $w_2 = v_2 - \langle v_2, e_1 \rangle e_1 \neq 0$.

Presupunem prin absurd că $w_2 = 0$. Atunci $v_2 = \langle v_2, e_1 \rangle e_1$, de unde rezultă că v_1 și v_2 sunt liniari dependenți. Contradicție!

Pentru w_2 și e_1 avem

$$\langle w_2, e_1 \rangle = \langle v_2, e_1 \rangle - \langle v_2, e_1 \rangle \langle e_1, e_1 \rangle = \langle v_2, e_1 \rangle - \langle v_2, e_1 \rangle = 0,$$

de unde rezultă $w_2 \perp e_1$. Fie $e_2 = \frac{1}{\|w_2\|} \cdot w_2 = \frac{1}{\|w_2\|} \cdot (v_2 - \langle v_2, e_1 \rangle e_1)$ pentru care avem $e_2 \perp e_1$ și e_2 unitar. Astfel $\{e_1, e_2\}$ este un sistem ortonormal; mai mult, $sp\{e_1, e_2\} = sp\{v_1, v_2\}$.

Pe scurt, am definit

$$\begin{cases} e_1 = \frac{1}{\|v_1\|} v_1 \\ e_2 = \frac{1}{\|w_2\|} w_2 - \frac{1}{\|w_2\| \|v_1\|} \langle v_2, e_1 \rangle v_1 \end{cases}.$$

Presupunem că am construit $\{e_1, \dots, e_k\}$, $k < n$ cu proprietatea că $\langle e_i, e_j \rangle = \delta_{ij}$, $\forall i, j = \overline{1, k}$ și

$$sp\{e_1, \dots, e_i\} = sp\{v_1, \dots, v_i\}$$

(spații cu aceeași orientare), $\forall i = \overline{1, k}$.

Definim $w_{k+1} = v_{k+1} - \sum_{i=1}^k \langle v_{k+1}, e_i \rangle e_i$. Se arată că $w_{k+1} \neq 0$ și $w_{k+1} \perp e_i$,

$\forall i = \overline{1, k}$. Construim $e_{k+1} = \frac{1}{\|w_{k+1}\|} \cdot w_{k+1}$ pentru care avem

$sp\{e_1, \dots, e_{k+1}\} = sp\{v_1, \dots, v_{k+1}\}$ (spații cu aceeași orientare) și pentru care rezultă că $\{e_1, \dots, e_{k+1}\}$ este sistem ortonormal.

Propoziție. Fie $\{e_1, \dots, e_k\}$ sistem ortonormat în $(V, \langle \cdot, \cdot \rangle)$. Presupunem $e_i \neq 0$, $\forall i = \overline{1, k}$. Atunci $\{e_1, \dots, e_k\}$ sistem de vectori liniari independenți.

Corolar. Orice spațiu vectorial euclidian finit admite o bază ortonormală.

Definiție. Fie V un spațiu vectorial peste \mathbb{R} și $g: V \times V \rightarrow \mathbb{R}$ o aplicație simetrică, biliniară și pozitiv definită (g este produs scalar). Atunci (V, g) se numește **spațiu vectorial Euclidian**.

XI. SPATIUL EUCLIDIAN \mathbb{R}^n

XI.1. Produs exterior

Fie $v_1, \dots, v_n \in \mathbb{R}^n$,

$$v_1 \wedge v_2 \wedge \dots \wedge v_n = \begin{vmatrix} v_1^1 & \cdots & v_1^n \\ \vdots & \ddots & \vdots \\ v_n^1 & \cdots & v_n^n \end{vmatrix} = \det(v_1, \dots, v_n).$$

Observații.

1. $v_1 \wedge \dots \wedge v_n = 0$ dacă și numai dacă $\{v_1, \dots, v_n\}$ sistem de vectori liniari dependenți;
2. Sistemul de vectori $\{v_1, \dots, v_n\}$ liniar independent este pozitiv orientat dacă și numai dacă $v_1 \wedge \dots \wedge v_n > 0$.

Definiție. Fie $v_1, \dots, v_p \in V$ (p este număr natural nenul arbitrar). Se numește **determinant Gram** asociat vectorilor v_1, \dots, v_p numărul

$$G(v_1, \dots, v_p) = \begin{vmatrix} \langle v_1, v_1 \rangle & \langle v_1, v_2 \rangle & \cdots & \langle v_1, v_p \rangle \\ \langle v_2, v_1 \rangle & \langle v_2, v_2 \rangle & \cdots & \langle v_2, v_p \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle v_p, v_1 \rangle & \langle v_p, v_2 \rangle & \cdots & \langle v_p, v_p \rangle \end{vmatrix}.$$

Observație. Fie $v_1, \dots, v_n \in V$ și $w_1, \dots, w_n \in V$, putem calcula

$$\begin{aligned} (v_1 \wedge \dots \wedge v_n)(w_1 \wedge \dots \wedge w_n) &= \begin{vmatrix} v_1^1 & \cdots & v_1^n \\ \vdots & \ddots & \vdots \\ v_n^1 & \cdots & v_n^n \end{vmatrix} \cdot \begin{vmatrix} w_1^1 & \cdots & w_1^n \\ \vdots & \ddots & \vdots \\ w_n^1 & \cdots & w_n^n \end{vmatrix} = \\ &= \begin{vmatrix} \langle v_1, w_1 \rangle & \cdots & \langle v_1, w_n \rangle \\ \vdots & \ddots & \vdots \\ \langle v_n, w_1 \rangle & \cdots & \langle v_n, w_n \rangle \end{vmatrix}. \end{aligned}$$

În particular pentru $w_i = v_i$, $(v_1 \wedge \dots \wedge v_n)^2 = G(v_1, \dots, v_n)$.

Pentru $n = 2$ (în planul \mathbb{R}^2 canonic) obținem

$$(v \wedge w)^2 = G(v, w) = \begin{vmatrix} \langle v, v \rangle & \langle v, w \rangle \\ \langle v, w \rangle & \langle w, w \rangle \end{vmatrix} = \|v\|^2 \cdot \|w\|^2 - \langle v, w \rangle^2$$

Presupunând că $v, w \neq 0$, $\cos \varphi = \frac{\langle v, w \rangle}{\|v\|\|w\|}$, rezultă

$(v \wedge w)^2 = \|v\|^2 \|w\|^2 (1 - \cos^2 \varphi)$, de unde $|v \wedge w| = \|v\| \cdot \|w\| \cdot |\sin \varphi|$ care este aria paralelogramului determinat de v și w .

XI.2. Produs vectorial

Fie $v_1, \dots, v_{n-1} \in \mathbb{R}^n$. Definim $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $f(x) = v_1 \wedge \dots \wedge v_{n-1} \wedge x$.

Atunci $f \in \text{Hom}(\mathbb{R}^n, \mathbb{R}) = (\mathbb{R}^n)^*$ (\mathbb{R}^n dual). Pentru $x, y \in \mathbb{R}^n$ și $a, b \in \mathbb{R}$ avem $f(ax + by) = v_1 \wedge \dots \wedge v_{n-1} \wedge (ax + by) = af(x) + bf(y)$.

Teorema Riesz. Fie $(V, \langle \cdot, \cdot \rangle)$ spațiu vectorial Euclidian n -dimensional, atunci există o corespondență bijectivă între V și V^* , $x \mapsto \langle x, \bullet \rangle$ (chiar izomorfism de spații vectoriale). În aceste condiții, formei liniare f definite mai sus îi corespunde un unic vector $v_1 \times \dots \times v_{n-1}$, care se numește **produs vectorial** al vectorilor v_1, \dots, v_n . Avem:

$$f(x) = \langle v_1 \times \dots \times v_{n-1}, x \rangle \Leftrightarrow v_1 \wedge \dots \wedge v_{n-1} \wedge x = \langle v_1 \times \dots \times v_{n-1}, x \rangle.$$

Observații.

1. $x \perp v_1 \times \dots \times v_{n-1}$ dacă și numai dacă $v_1 \wedge \dots \wedge v_{n-1} \wedge x = 0$. Presupunem $x \neq 0$ (cazul $x = 0$ este evident); presupunem $\{v_1, \dots, v_{n-1}\}$ sistem liniar independent. Notăm $V = sp\{v_1, \dots, v_{n-1}\}$ și este subspațiu vectorial $(n-1)$ -dimensional. Atunci $x \perp v_1 \times \dots \times v_{n-1}$ dacă și numai dacă $x \notin V$. Rezultă $v_1 \times \dots \times v_{n-1} \perp V$ dacă și numai dacă $v_1 \times \dots \times v_{n-1} \in V^\perp$
2. Pentru $x = v_1 \times \dots \times v_{n-1}$, rezultă

$$v_1 \wedge \dots \wedge v_{n-1} \wedge (v_1 \times \dots \times v_{n-1}) = \|v_1 \times \dots \times v_{n-1}\|$$

3. $v_1 \times \dots \times v_{n-1} = 0$ dacă și numai dacă $\{v_1, \dots, v_{n-1}, v_1 \times \dots \times v_{n-1}\}$ este sistem de vectori liniar dependenți.

Propoziție. Produsul vectorial este antisimetric.

Observație (Produs vectorial dublu). Are loc relația

$$(x \times y) \times z = \langle x, z \rangle y - \langle y, z \rangle x.$$

Considerăm că $x = (x^1, x^2, x^3)$, $y = (y^1, y^2, y^3)$ și $z = (z^1, z^2, z^3)$.

Putem scrie $\langle x, z \rangle = x^1 z^1 + x^2 z^2 + x^3 z^3$ și $\langle y, z \rangle = y^1 z^1 + y^2 z^2 + y^3 z^3$.

Înlocuind se obține

$$\langle x, z \rangle y - \langle y, z \rangle x = (+x^2 y^1 z^2 + x^3 y^1 z^3 - x^1 y^2 z^2 - x^1 y^3 z^3, \dots, \dots).$$

De asemenea,

$$\begin{vmatrix} x^1 & x^2 & x^3 \\ y^1 & y^2 & y^3 \\ \rightarrow & \rightarrow & \rightarrow \\ i_1 & i_2 & i_3 \end{vmatrix} = A^1 \xrightarrow{i_1} + A^2 \xrightarrow{i_2} + A^3 \xrightarrow{i_3} = (A^1, A^2, A^3) = \\ = (x^2 y^3 - x^3 y^2, -x^1 y^3 + x^3 y^1, x^1 y^2 - x^2 y^1) = x \times y$$

Înlocuind și efectuând calculele, obținem

$$\begin{vmatrix} x^2 y^3 - x^3 y^2 & -x^1 y^3 + x^3 y^1 & x^1 y^2 - x^2 y^1 \\ z^1 & z^2 & z^3 \\ \rightarrow & \rightarrow & \rightarrow \\ i_1 & i_2 & i_3 \end{vmatrix} = \\ = (+x^2 y^1 z^2 + x^3 y^1 z^3 - x^1 y^2 z^2 - x^1 y^3 z^3, \dots, \dots)$$

Propoziție (Identitatea lui Iacobi).

$$(x \times y) \times z + (y \times z) \times x + (z \times x) \times y = 0.$$

Observație. Produsul vectorial este aplicație \mathbb{R} -biliniară, adică $\forall a, b \in \mathbb{R}, \forall x_1, x_2, y \in \mathbb{R}^3$,

$$(ax_1 + bx_2) \times y = a(x_1 \times y) + b(x_2 \times y).$$

XII. MORFISME DE SPAȚII VECTORIALE EUCLIDIENE

Fie $(V, \langle \cdot, \cdot \rangle)$, (W, g) spații vectoriale euclidiene și $t : V \rightarrow W$ morfism de spații vectoriale (adică $t \in \text{Hom}(V, W)$).

Definiție. t se numește **morfism de spații vectoriale euclidiene (izometrie liniară, aplicație ortogonală)** dacă $\forall v_1, v_2 \in V$,

$$g(t(v_1), t(v_2)) = \langle v_1, v_2 \rangle \quad (1)$$

Propoziție. În condițiile de mai sus următoarele afirmații sunt echivalente:

- i) Are loc relația (1)
- ii) $\|t(v)\|_g = \|v\|_{\langle \cdot, \cdot \rangle}, \forall v \in V$.

Observație. $t : V \rightarrow W$ este aplicație ortogonală dacă și numai dacă păstrează lungimea vectorilor.

Corolar. Fie $t : V \rightarrow W$ aplicație ortogonală. Atunci t păstrează unghiiurile (reciproca este falsă).

Observație. Orice aplicație ortogonală este injectivă. Știm că $t \in \text{Hom}(V, W)$ este injectivă dacă și numai dacă $\text{Ker } t = \{0_V\}$. Fie $x \in \text{Ker } t$ de unde $t(x) = 0$, astfel încât $\|t(x)\| = 0$, ceea ce implică $\|x\| = 0$ și rezultă $x = 0$.

Corolar. O aplicație ortogonală $t : V \rightarrow W$ este izomorfism de spații vectoriale dacă și numai dacă t este surjectivă. În acest caz $t^{-1} : W \rightarrow V$ este tot aplicație ortogonală.

Observații.

1. V este izomorf cu W dacă și numai dacă $\dim V = \dim W$.
2. Fie $V = W$, $g = \{\cdot, \cdot\}$, $\dim V = n$, $\{e_1, \dots, e_n\}$ o bază ortonormală a lui V . Fie $t : V \rightarrow V$ aplicație ortogonală astfel încât pentru

$$x \in V, x = x^1 e_1 + \dots + x^n e_n = \begin{pmatrix} x^1 \\ \vdots \\ x^n \end{pmatrix}$$

să avem $t(x) = \sum x^i t(e_i) = \sum x^i t_i^j e_j$. Formăm matricea

$$T = (t_i^j)_{i,j=1,n} \in \mathbf{M}_{n,n}(\mathbb{R}).$$

Putem face următoarele afirmații:

- a) $T \in GL(n, \mathbb{R})$ (deoarece t este izomorfism de spații vectoriale);
- b) $\det T \neq 0$;
- c) următoarele relații sunt echivalente

$$t(x) = Tx \quad (x \text{ vector coloană}) \quad (1)$$

$$\langle Tx, Ty \rangle = \langle x, y \rangle, \quad \forall x, y \in V \quad (1')$$

d)

$${}^t TT = I_n \quad (2)$$

Propoziție. $t : V \rightarrow V$ aplicație ortogonală dacă și numai dacă are loc (2)

Corolar. Dacă are loc relația 2, atunci $T^{-1} = {}^t T$.

Notăm cu $o(n) = \{T \in GL(n, \mathbb{R}) \mid {}^t TT = I_n\}$. Atunci $o(n)$ este subgrup al lui $GL(n, \mathbb{R})$ și se numește **grupul ortogonal**.

Propoziție.

1. Dacă $T \in o(n)$, atunci $\det T = \pm 1$.
2. $So(n) = \{T \in o(n) \mid \det T = 1\}$ este subgrup al lui $o(n)$ și se numește **grupul special ortogonal**.

Observație. Fie V spațiu vectorial euclidian n -dimensional, $\{e_1, \dots, e_n\}$ bază ortonormală, $t : V \rightarrow V$ aplicație ortogonală având $T \in o(n)$. Fie $x \in V$ vector propriu ($\exists \lambda \in \mathbb{R}, t(x) = \lambda x$). Atunci $\lambda = \pm 1$ (din $\|t(x)\| = \|x\|$, obținem $|\lambda| \cdot \|x\| = \|x\|$, de unde $|\lambda| = 1$).

Propoziție. Valorile proprii ale unei aplicații ortogonale sunt ± 1 .

Observație. Fie x un vector propriu corespunzător valorii proprii λ , nenul. $x^\perp = \{v \in V | v \perp x\}$ și știm că $\dim x^\perp = n - 1$. Avem $v \perp x$, dacă și numai dacă $\langle v, x \rangle = 0$, dacă și numai dacă $\langle t(v), t(x) \rangle = 0$. Cum $t(x) = \lambda x$, atunci $\lambda \langle t(v), x \rangle = 0$, de unde $\langle t(v), x \rangle = 0$. Rezultă $t(v) \perp x$, de unde $t(v) \in x^\perp$

Propoziție. t invariază pe x^\perp .

Exemplu.

1. Simetria, $t : V \rightarrow V$, $t(v) = -v$. Avem $\langle t(v), t(v) \rangle = \langle -v, -v \rangle = \langle v, v \rangle$
2. Fie $\theta \in \mathbb{R}$, $t : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $e_1 = (1,0)$, $e_2 = (0,1)$ și $T = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$.

Avem $\det T = 1$. $T \in o(2)$ dacă și numai dacă $'TT = I_2$ (ceea ce se verifică prin calcul). Notăm $T = T_\theta \in So(2)$.

XIII. APLICAȚII SIMETRICE

Fie $(V, \langle \cdot, \cdot \rangle)$ un spațiu vectorial euclidian n -dimensional. Peste \mathbb{R} considerăm spațiul vectorial

$$L(V) = \text{End}(V) = \{f : V \rightarrow V | f \text{ este } R\text{-liniar}\} \subset \text{Hom}(V, V)$$

Definiție. Fie $t \in L(V)$. t se numește **morfism simetric** în $(V, \langle \cdot, \cdot \rangle)$ dacă $\langle t(x), y \rangle = \langle x, t(y) \rangle$.

Propoziție. $L^S(V) = \{f \in L(V) | f \text{ simetrică}\} \subset L(V)$ este subspațiu vectorial.

Observație. Fixăm o bază ortonormală $\{e_1, \dots, e_n\}$ a lui $(V, \langle \cdot, \cdot \rangle)$. Știm că există o bijecție între $L(V)$ și $\mathbf{M}_{n,n}(\mathbb{R})$, adică avem $t(e_i) = t_i^j e_j$, $i = \overline{1, n}$, $T = (t_i^j)_{i,j=\overline{1,n}} \in \mathbf{M}_{n,n}(\mathbb{R})$.

Propoziție. În condițiile de mai sus, $t \in L^S(V)$ dacă și numai dacă $T = 'T$.

Observație. Fie $x \in V$ vector propriu corespunzător valorii proprii λ pentru un morfism $t \in L^S(V)$, $t(x) = \lambda x$, $\lambda \in \mathbb{C}$. Notăm cu x^\perp complementul ortogonal al lui x . Atunci $V = R \oplus x^\perp$.

Propoziție. În condițiile de mai sus, $t \in L^S(V)$ invariază complementul ortogonal al oricărui vector propriu al său.

Observație. Valorile proprii ale unui endomorfism $t \in L(V)$ se determină calculând rădăcinile polinomului caracteristic asociat. Fixăm o bază (ortonormală) și

$T \in \mathbf{M}_{n,n}(\mathbb{R})$ asociata lui t . **Polinomul caracteristic asociat** este $\det(T - \lambda I_n) = 0$ (ecuatie polinomiala de grad n cu necunoscuta λ). Rădăcinile sunt numere complexe (în particular pot fi reale).

Propoziție. Fie t un morfism simetric al lui V ; $t \in L^S(V)$. Atunci valorile proprii ale lui t sunt reale.

XIV. SPAȚII AFINE EUCLIDIENE

Definiție. Fie (ξ, V, φ) un spațiu afin de spațiu director $(V, \langle \cdot, \cdot \rangle)$, acesta fiind un spațiu vectorial euclidian. Spunem că ξ se numește **spațiu euclidian**.

Exemplu. \mathbb{R}^n cu structura canonică de spațiu afin și $\varphi(P, Q) = Q - P$.

Observație. Fie ξ un spațiu euclidian de spațiu vectorial director $(V, \langle \cdot, \cdot \rangle)$. Fie $P, Q \in \xi$ și $\varphi(P, Q) = PQ = \overrightarrow{PQ} = \vec{PQ} \in V$.

Definiție. Se numește **distanța** între P și Q în ξ , $d(P, Q) = \|\overrightarrow{PQ}\|$.

Proprietăți.

1. $d(P, Q) = d(Q, P)$, $\forall P, Q \in \xi$;
2. $d(P, Q) \leq d(P, S) + d(S, Q)$, $\forall P, Q, S \in \xi$;
3. $d(P, Q) \geq 0$, $\forall P, Q \in \xi$;
4. $d(P, Q) = 0 \Leftrightarrow P = Q$.

Exemplu. Fie \mathbb{R}^n cu structura canonică de spațiu euclidian și $\{e_1, \dots, e_n\}$

baza ortonormală, $\begin{cases} e_1 = (1, 0, \dots, 0) \\ \vdots \\ e_n = (0, \dots, 0, 1) \end{cases}$. Reperul cartezian ortonormal este $\{0; e_1, \dots, e_n\}$. Fie $P(p^1, \dots, p^n)$ și $Q(q^1, \dots, q^n)$. Avem $\overrightarrow{PQ} = Q - P = (q^1 - p^1, \dots, q^n - p^n)$, de unde

$$d(P, Q) = \|\overrightarrow{PQ}\| = \sqrt{(q^1 - p^1)^2 + \dots + (q^n - p^n)^2}.$$

Definiție. Fie ξ un spațiu euclidian n -dimensional ca mai sus, A_0 un punct și $v_1, \dots, v_p \in V$ liniari independenți. Se numește **paralelipiped generat de** A_0 și **de vectorii** v_1, \dots, v_p mulțimea

$$[A_0; v_1, \dots, v_p] = \{P \in \xi \mid A_0 P = \alpha^1 v_1 + \dots + \alpha^p v_p\}, \quad \forall \alpha^i \in [0,1].$$

Se numește **simplex generat de** A_0 și **vectorii** v_1, \dots, v_p mulțimea

$$[A_0; A_1, \dots, A_p] = \{P \in \xi \mid P = \beta^0 A_0 + \beta^1 A_1 + \dots + \beta^p A_p\}$$

cu $\beta^0, \dots, \beta^p \in [0,1]$, $\beta^0 + \dots + \beta^p = 1$.

Definiție. Ca mai sus, fie $[A_0; v_1, \dots, v_p]$ paralelipiped și $[A_0; A_1, \dots, A_p]$ simplexul asociat. **Volumul paralelipipedului** se definește prin

$$V[A_0; v_1, \dots, v_p] = \sqrt{G(v_1, \dots, v_p)}$$

Volumul simplexului este dat de relația

$$V[A_0; A_1, \dots, A_p] = \frac{1}{p!} \sqrt{G(v_1, \dots, v_p)}$$

unde

$$G(v_1, \dots, v_p) = \begin{vmatrix} \langle v_1, v_1 \rangle & \cdots & \langle v_1, v_p \rangle \\ \vdots & \ddots & \vdots \\ \langle v_p, v_1 \rangle & \cdots & \langle v_p, v_p \rangle \end{vmatrix}.$$

Exemplu. Pentru $p = 1$, avem

$$V[A_0; v_1] = V[A_0; A_1] = \sqrt{G(v_1)} = \sqrt{\langle v_1, v_1 \rangle} = \|v_1\| = \|\overrightarrow{A_0 A_1}\|.$$

Pentru $p = 2$ avem

$$\begin{aligned} V[A_0; v_1, v_2] &= \sqrt{\begin{vmatrix} \langle v_1, v_1 \rangle & \langle v_1, v_2 \rangle \\ \langle v_2, v_1 \rangle & \langle v_2, v_2 \rangle \end{vmatrix}} = \sqrt{\|v_1\|^2 \|v_2\|^2 - \langle v_1, v_2 \rangle^2} = \\ &= \sqrt{\|v_1\|^2 \|v_2\|^2 - \cos^2 \theta \|v_1\|^2 \|v_2\|^2} = \|v_1\| \cdot \|v_2\| \cdot |\sin \theta| = \\ &= 2V[A_0; A_1, A_2] \end{aligned}$$

Pentru $n = 2$, $p = 2$ avem

$$V[A_0; v_1, v_2] = \pm \begin{vmatrix} 1 & 1 & 1 \\ x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \end{vmatrix}.$$

Pentru $n = 3$, $p = 3$ avem

$$V[A_0; v_1, v_2, v_3] = \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \end{vmatrix}.$$

BIBLIOGRAFIE

1. Craioveanu M., Albu I.D., *Geometrie afină și euclidiană*, Editura Facla, Timișoara, 1982.
2. Teleman K., *Logică și geometrie*, Imprimăria Universității București, 1989.
3. Turtoi A., *Geometrie*, Tipografia Universității București, 1983.

ALGORITMI ȘI PROGRAMARE

Prof. univ dr. GRIGORE ALBEANU
Lector univ. SILVIU BÂRZĂ

I. INTRODUCERE

1.1. Ce este informatica?

În anul 1642, matematicianul și fizicianul Blaise Pascal (1623-1662) a inventat prima mașină mecanică, cu roți dințate, capabilă să realizeze operații de adunare și scădere a numerelor naturale. Totuși, de-abia după apariția mașinilor electromecanice, în 1944, John von Neumann a formulat principiul programului înregistrat și a sugerat constructorilor de calculatoare trei principii care trebuie avute în vedere pentru realizarea unui calculator:

- programele și datele trebuie să fie codificate sub formă binară;
- programele și datele trebuie stocate într-o memorie a mașinii de calcul;
- trebuie să existe o componentă (unitate centrală de prelucrare, procesor) specială care știe atât să execute operații de calcul, cât și să extragă, să decodifice și să execute instrucțiunile programului.

Astfel, aproape toate tipurile de sisteme de calcul ce au apărut mai târziu sunt calculatoare de tip von Neumann.

Apariția sistemelor de calcul a dus la apariția și dezvoltarea unei noi științe: informatica (*Informatique*, *Informatics*, *Informatik* în Europa, respectiv *Computer Science* în SUA). Informatica reprezintă un complex de discipline prin care se asigură prelucrarea informațiilor cu ajutorul sistemelor de calcul. Astăzi, informatica este prezentă în industrie, bănci, ferme, artă, medicină, științe sociale etc. și este structurată în mai multe domenii precum:

- *arhitectura sistemelor de calcul*: studiază modul de organizare a sistemului fizic (*hardware*) pentru a se obține o mai mare eficiență, siguranță și utilitate.
- *sisteme de operare*: studiază modalitățile de gestiune eficientă a resurselor fizice și a programelor.
- *algoritmi și structuri de date*: studiază metodele de rezolvare a problemelor și modurile de organizare a datelor pentru a obține programe eficiente.
- *limbaje de programare*: studiază modalitățile prin care algoritmii și structurile de date sunt prezentate calculatorului pentru a fi prelucrate.
- *ingineria programării*: studiază metodele de automatizare a proceselor de proiectare, analiză, testare și reutilizare a programelor.
- *calcule numerice și simbolice*: studiază modalitățile de reprezentare a informației numerice pentru implementarea unor algoritmi numerici robusti, eficienți și de mare precizie.

- *sisteme de gestiune a bazelor de date*: studiază modalitățile de structurare și organizare eficientă a colecțiilor mari de date ce vor fi supuse diverselor prelucrări.
- *inteligenta artificială*: studiază modalitățile de reprezentare și manipulare a cunoștințelor în vederea obținerii de noi cunoștințe.
- *animația și robotica*: studiază modalitățile de reprezentare, prelucrare și analiză a informației audio-vizuale.

În sfera sa de preocupări, informatica a atras și dezvoltat o mare varietate de discipline precum: logica matematică, teoria automatelor, limbaje formale, cercetări operaționale, teoria grafurilor și rețelelor, calcul numeric, teoria fiabilității, geometrie computațională, teoria calculabilității, baze de date, baze de cunoștințe, sisteme expert .

I.2. Ce este un program? Noțiunea de algoritm

Soluția unei probleme, din punct de vedere informatic, este dată printr-o mulțime de comenzi (instrucțiuni) explicite și neambigue, exprimate într-un limbaj de programare. Această mulțime de instrucțiuni prezentată conform anumitor reguli sintactice formează un program. Un program poate fi privit și ca un algoritm exprimat într-un limbaj de programare. Totuși, un algoritm descrie soluția problemei independent de limbajul de programare în care este redactat programul.

Există mai multe definiții ale noțiunii de algoritm. A. A. Markov, a considerat algoritmul ca fiind o noțiune matematică primară și a descris-o astfel: *Un algoritm este o rețetă care descrie precis și clar un proces de calcul*. El a descris un mecanism formal pentru a specifica o clasă largă de activități și anume: algoritmii normali. Alte modalități de descriere a algoritmilor ce au mai fost propuse sunt: mașina Turing, sistemele Post, funcțiile recursive etc. În această lucrare, prin algoritm vom înțelege o secvență finită de comenzi explicite și neambigue care, executate pentru o mulțime de date (ce satisfac anumite condiții initiale), conduce în timp finit la rezultatul corespunzător. Observăm că se face distincție între algoritm (care se termină în timp) și procedură (care poate continua nedefinit).

Conform lui D. Knuth (*The art of computer programming*, Vol. I, 1997) un algoritm posedă cinci caracteristici importante:

- *Un algoritm are caracter finit*. El este descris printr-o secvență finită de etape și trebuie ca, ori de câte ori sunt parcuse etapele algoritmului pentru anumite date, procesul să se încheie în timp finit.
- *Un algoritm are caracter determinist*. Acțiunile specificate de pașii algoritmului sunt clare, fiind riguroș prezентate.
- *Un algoritm are date de intrare*. Se poate admite că întotdeauna un algoritm lucrează asupra unor date de intrare. Acestea pot fi evidențiate din contextul în care este formulată problema a cărei soluție o reprezintă algoritmul.
- *Un algoritm furnizează cel puțin o valoare de ieșire ce se află într-o relație specifică cu datele de intrare*.
- *Un algoritm este eficace*.

Trebuie spus că nu orice problemă admite soluție descrisă algoritmic. Problemele pentru care există un algoritm de rezolvare se numesc *probleme decidabile*. Problemele pentru care s-a demonstrat (matematic) că nu admit un algoritm de rezolvare se numesc *probleme nedecidabile*. Nu este suficient ca o anumită problemă (clasă de probleme) să admită soluții (chiar și o singură soluție). Din punctul de vedere al informaticii, interesează dacă există soluție ce poate fi descrisă algoritmic, deci dacă soluția problemei poate fi construită efectiv. De asemenea, pentru rezolvarea anumitor probleme pot exista mai mulți algoritmi. Stabilirea celui mai bun dintre aceștia se realizează în urma unui proces de analiză prin care se determină performanțele fiecărui dintre algoritmi.

Am văzut că algoritmii acceptă date de intrare și furnizează rezultate (date de ieșire). Introducerea unei valori sau a unui sir de valori se va descrie folosind verbul **citește**. Afisarea unei valori sau a unui sir de valori va fi descrisă prin verbul **scrive**. Vom conveni să numerotăm pașii (etapele) unui algoritm. Această numerotare va fi utilizată eventual în alți pași. Acest mod de prezentare a algoritmilor se numește descriere în limbaj convențional. În unele lucrări, limbajul convențional mai este numit și limbaj pseudocod.

Exemplul 1. Fie a și b două variabile întregi. Dorim să schimbăm între ele valorile celor două variabile. Mai precis, dacă $a = 640$ și $b = 480$, dorim să obținem $a = 480$ și $b = 640$.

Un mod de a realiza această schimbare presupune utilizarea unei variabile suplimentare, cu rol de intermediar. Folosind trei operații de atribuire, algoritmul are următorii pași:

1. citește a, b
2. $t := a$
3. $a := b$
4. $b := t$
5. scrie a, b

I.3. Cum rezolvăm probleme cu ajutorul calculatorului?

Am văzut că există probleme pentru care nu poate fi dat un algoritm de rezolvare. Totuși cele mai multe probleme cu care se confruntă informatica sunt probleme decidabile. Toate temele tratate în această lucrare formulează probleme decidabile.

Se știe că nu învățarea unui limbaj de programare este o sarcină dificilă, ci rezolvarea algoritmică a problemelor decidabile. Este clar că, înainte de a începe scrierea unui program trebuie, mai întâi, să găsim (sau să cunoaștem deja) soluția algoritmică a problemei puse. Cum se găsește soluția? Etapele descrise de G. Pólya (*Cum rezolvăm o problemă?* Editura Științifică, București, 1965), pentru rezolvarea unei probleme de matematică, sunt valabile și în informatică.

Înainte de a începe să vedem cum se face, trebuie să înțelegem atât ipotezele problemei, cât și ceea ce se cere. Deci, înțelegerea problemei, ocupă primul loc în procesul căutării unei metode de rezolvare a acesteia cu ajutorul calculatorului. Trebuie evidențiate datele de intrare și condițiile pe care acestea

trebuie să le îndeplinească. Este important să identificăm esențialul. De multe ori un enunț al unei probleme conține foarte multe elemente descriptive privind importanța problemei, considerații de natură istorică, exemple, uneori chiar și etape distincte pentru obținerea soluției. Cerința problemei furnizează, de multe ori, chiar idei privind modul de a ajunge la soluție. Considerarea unor configurații particulare pentru datele de intrare și încercarea de a lucra asupra lor, pentru a găsi soluția, va contribui întotdeauna la o înțelegere mai bună a enunțului.

Dacă în enunț se sugerează etapele rezolvării, acestea implicând rezolvarea unor subprobleme, atunci trebuie să aflăm soluția algoritmică corespunzătoare fiecărei etape. Dacă nu se specifică etape, dar putem descompune problema în subprobleme mai simple, atunci soluția algoritmică a problemei inițiale se va obține prin "componerea" soluțiilor subproblemelor. Deci, este foarte important să știm să rezolvăm probleme simple, eventual probleme reprezentative pentru anumite clase de aplicații și să știm să descompunem (respectiv, să reducem) problema inițială în (la) subprobleme ușor de rezolvat și apoi să construim soluția finală. Abordarea prin descompuneri repetitive, cu detaliere pas cu pas se numește abordare *top-down* sau *refinare iterativă*. Abordarea prin care, pornind de la soluții algoritmice ale unor probleme cunoscute, construim soluții ale altor probleme care au însă legătură cu problema de rezolvat, iar în final, urmând aceeași modalitate construim soluția problemei a cărei soluție se cere, se numește abordare *bottom-up*. Această metodă permite reutilizarea programelor existente și este tot mai importantă odată cu apariția tehniciilor orientate obiect. De asemenea, pentru a obține o soluție a unei probleme este util să privim problema și comparativ cu alte probleme întâlnite. Numai după investigarea acestor variante putem trece la stabilirea metodei de abordare.

Pentru probleme de complexitate ridicată, oricare din metodele de abordare *top-down* sau *bottom-up*, conduc la o soluție modulară, bazată pe *subprograme* sau, mai nou, la o *soluție orientată obiect*.

Multe din problemele de programare pot fi rezolvate ușor dacă soluția verifică anumite principii. Dacă soluția problemei este o mulțime de elemente care se poate obține pas cu pas, pornind de la mulțimea vidă, prin adăugarea celui mai bun element neconsiderat încă (și ales conform unui anumit criteriu) spunem că avem de-a face cu o abordare de tip *greedy*. Pentru probleme în care se cere o soluție optimă care satisfacă principiul optimalității (principiul lui Bellman) se va aplica *metoda programării dinamice*. Alte strategii pentru elaborarea algoritmilor sunt: *metoda divide et impera*, *metoda backtracking*, *euristica* etc.

II. ALGORITMI: DESCRIERE, COMPLEXITATE, CORECTITUDINE

II.1. Noțiuni de teoria grafurilor

Un graf neorientat G este definit prin perechea $G = (V, E)$, unde V este o mulțime nevidă de elemente numite vârfuri (vertex), iar E este o mulțime (posibil vidă) de perechi neordonate cu componente distințe ale lui V care se numesc muchii (edges). Dacă E este mulțimea vidă spunem că G este trivial. În cazul în care mulțimea V este finită spunem că avem un graf finit. Numărul elementelor mulțimii V determină ordinul grafului finit. O muchie cu vârfurile x și y (numite extremități) se notează prin $[x, y]$ sau $[y, x]$. Spunem că vârfurile x și y sunt incidente muchiei $[x, y]$.

Un digraf este definit printr-o pereche $G = (V, E)$, unde V este o mulțime nevidă de vârfuri, iar E este o parte a produsului cartezian $V \times V$ ale cărei elemente le numim arce. Un arc este notat prin (x, y) cu x diferit de y , unde x se numește sursă sau extremitate inițială, iar y se numește destinație sau extremitate finală. Atributul : trivial, respectiv finit, pentru un digraf se definesc similar.

Un graf (digraf) parțial al unui graf (digraf) $G = (V, E)$ este un graf (digraf) $G_1 = (V, E_1)$, unde E_1 este submulțime a mulțimii E , deci este graful (digraful) G însuși sau se obține din G prin suprimarea anumitor muchii (arce).

Un subgraf (subdigraf) al unui graf (digraf) G este un graf (digraf) $H = (U, E')$, unde U este submulțime a mulțimii V , E' este submulțime a mulțimii E , iar muchiile (arcele) din E' sunt toate muchiile (arcele) din E , care au ambele extremități în mulțimea de vârfuri U .

Uneori, arcele (muchiile) sunt etichetate. Dacă etichetele sunt numere reale pozitive se spune că digraful (graful) este ponderat.

În continuare vom considera numai grafuri (digrafuri) finite. De obicei un graf se reprezintă prin indicarea unor puncte din plan (ce corespund vârfurilor) și a unor segmente de curbă (sau de dreaptă) pentru indicarea muchiilor. În cazul digrafurilor arcele sunt segmente de curbă orientate, sensul fiind de la sursă spre destinație.

Definiție. Fie $G = (V, E)$ un digraf, iar x și y două vârfuri. Numim x - y drum (de lungime n) o secvență de vârfuri $D: v_0, v_1, \dots, v_n$ dacă $v_0 = x$, $v_n = y$, iar (v_i, v_{i+1}) aparține mulțimii E pentru toți i , $1 \leq i \leq n-1$. Vârfurile x și y se numesc extremitățile drumului D . Un drum fără nici un arc este un drum trivial. Un x - y drum se numește circuit (sau drum închis) dacă $x=y$; dacă x diferit de y , se spune că drumul este unul deschis. Circuitul este elementar dacă toate vârfurile circuitului, cu excepția primului și a ultimului vârf care coincid, sunt distințe două câte două.

Definiție. Fie $G = (V, E)$ un graf neorientat, iar x și y două vârfuri. Secvența de vârfuri $L: v_0, v_1, \dots, v_n$ este un x - y lanț (de lungime n) dacă $[v_i, v_{i+1}]$ aparține mulțimii E , pentru toți i , $0 \leq i \leq n-1$, iar $x=v_0$ și $y=v_n$. L este ciclu dacă $x=y$. Atributul elementar, pentru un lanț, poate fi introdus similar.

Uneori, chiar pentru un digraf, putem folosi noțiunea de lanț, dacă se verifică proprietatea că oricare două arce vecine au o extremitate comună.

Definiție. Numim lanț (drum) hamiltonian un lanț (drum) elementar al unui graf care conține toate vârfurile grafului.

Definiție. Fie $G = (V, E)$ un graf netrivial. Spunem că x, y din V sunt conectate în G dacă există un $x-y$ lanț în G . Graful G este un graf conex dacă oricare două vârfuri din V sunt conectate în G . Dacă G este un graf trivial (E mulțime vidă) se acceptă că este graf conex. Dacă G nu este conex, atunci există cel puțin două componente conexe (subgrafuri conexe maximale, disjuncte două câte două relativ la vârfuri). Un digraf G cu proprietatea că pentru oricare două vârfuri x și y există atât un $x-y$ drum, cât și un $y-x$ drum în G se numește graf tare conex.

În secțiunea următoare prezentăm o metodă de reprezentare a algoritmilor folosind schemele logice. Acestea sunt introduse folosind terminologia teoriei grafurilor.

II.2. Scheme logice structurate

O transcriere grafică a etapelor (pașilor) unui algoritm este numită organigramă. De cele mai multe ori este folosită denumirea de "schemă logică". Fiecărui pas, al algoritmului, i se asociază un bloc ce constituie eticheta unui arc. Blocurile folosite într-o schemă logică descriu instrucțiuni (comenzi) sau predicate (expresii logice). Predicatelor apar în cadrul instrucțiunii de ramificare. Celelalte instrucțiuni sunt:

1. Instrucțiunea START: etichetează arcul inițial (acel arc în a căruia extremitate inițială nu pot sosi alte arce). Orice schemă logică are un unic arc inițial. Aceasta va indica punctul de unde începe execuția unui program.
2. Instrucțiunea STOP: etichetează un arc final (acel arc din a căruia extremitate finală nu pot pleca alte arce). O schemă logică poate avea mai multe arce finale. Acestea indică oprirea execuției programului descris prin intermediul schemei logice.
3. Instrucțiunea CTESTE: etichetează un arc ce indică introducerea de la mediul de intrare (tastatură, disc etc.) a unei secvențe de valori (numite date de intrare). Cuvântul CTESTE este însoțit de variabilele ce descriu datele de intrare.
4. Instrucțiunea SCRIE: etichetează un arc ce indică înregistrarea la mediul de ieșire (display, disc etc.) a rezultatelor (datelor de ieșire). Cuvântul SCRIE este însoțit de variabilele sau constantele ce desemnează datele de ieșire.
5. Instrucțiunea de atribuire: etichetează un arc cu eticheta $v := e$, unde v este o variabilă, iar e este o expresie de același tip (numeric sau logic) cu variabila v .
6. Instrucțiunea de ramificare: este caracterizată prin n arce ce pleacă din același punct, arce etichetate cu predicatele p_1, p_2, \dots, p_n definite astfel încât
$$p_1 \text{ or } p_2 \text{ or } \dots \text{ or } p_n = \text{TRUE} \text{ (adevărat)}$$
și
$$p_i \text{ and } p_j = \text{FALSE} \text{ (fals) pentru oricare } i \text{ și } j \text{ diferiți.}$$

Definiție. Se numește schemă logică (sau program sub formă de schemă logică) un graf orientat, în care:

- există o unică instrucțiune START și cel puțin o instrucțiune STOP;
- orice vârf (diferit de extremitatea finală a unei instrucțiuni STOP) este extremitatea inițială a unei unice instrucțiuni;
- orice arc este etichetat cu una dintre instrucțiunile: START, STOP, CITESTE, SCRIE, atribuire sau cu un predicat. În ultimul caz, extremitatea inițială a arcului trebuie să coincidă cu extremitatea inițială a unei instrucțiuni de ramificare;
- pentru orice arc există cel puțin un drum care începe cu instrucțiunea START, se termină cu o instrucțiune STOP și conține arcul considerat.

Schemele logice sunt folosite pentru descrierea algoritmilor. Se pot pune în evidență structuri fundamentale precum:

1. structura secvențială - formată din arce conectate etichetate cu instrucțiuni distincte de cea de ramificare. O structură secvențială formată din două arce etichetate prin a, respectiv b se va nota prin SEQ(a,b) și are semnificația "execută a urmat de b".

2. structuri decizionale (alternative) - conțin, obligatoriu, cel puțin un predicat și cel puțin un bloc funcțional (instrucțiune alta decât START sau ramificativ). Structurile decizionale sunt de următoarele forme:

- IF(p; a, b) - Dacă p este adevărat, atunci execută a altfel execută b.
- IF0(p; a) - Dacă p este verificat, atunci a.
- CASE(p₁,p₂,...,p_n; a₁, a₂, ..., a_n) - Dacă p₁ atunci a₁, dacă p₂ atunci a₂, ..., dacă p_n atunci a_n.

3. Structuri repetitive (structuri de tip ciclu, structuri iterative) - conțin obligatoriu un bloc predicativ și un bloc funcțional care se execută de un număr finit de ori până când predicatul își schimbă valoarea. Sunt posibile trei situații:

- WHILE(p; a) - Cât timp condiția p este adevărată se execută a, apoi se continuă cu următoarea instrucțiune.
- REPEAT(p; a) - Repetă a până când condiția p este adevărată, apoi execută instrucțiunea următoare.
- FOR(p; a, b, c) - Este echivalentă cu SEQ(a, WHILE(p; SEQ(b, c))). Blocul a este un bloc funcțional de initializare. Blocul b descrie instrucțiunea ce se executa când condiția p este adevărată. Blocul c (prezentat explicit în această structură) va descrie actualizarea stărilor variabilelor programului cu rol deosebit în evaluarea condiției p.

Etiqueta secvenței de mai sus este ET1, iar componentele secvenței sunt instrucțiuni Pascal. Secvența descrie citirea a trei numere, calculul mediei aritmetice și afișarea rezultatului.

Definiție. Un algoritm exprimat în funcție de structurile SEQ, IF și WHILE se numește structurat, schema logică asociată se numește schemă logică structurată, iar programul corespunzător se numește program structurat.

Evident, familia algoritmilor structurați este nevidă. Un rezultat foarte important afirmă: *Orice algoritm poate fi transformat într-un algoritm structurat.* Această afirmație are la bază teorema Bohm-Jacopini. Pentru a transforma o

schemă logică (nestructurată) într-o schemă logică structurată se poate folosi regula variabilei booleene (ce rezultă din demonstrația teoremei Bohm-Jacopini). Cititorul interesat de detalii poate consulta lucrarea: *Corrado Bohm, Giuseppe Jacopini - Flow-diagrams, Turing Machines and languages with only two formation rules. Comm. ACM. 9 (May, 1966), 366-371*. Una din consecințele importante ale programării structurate este eliminarea instrucțiunii de salt necondiționat (*goto*) din programe. Totuși, există situații când instrucțiunea *goto* este utilă. Lungimea programelor nu va crește, iar claritatea algoritmului nu va avea de suferit. Important este ca numărul instrucțiunilor *goto* folosite să fie foarte mic, iar salturile să fie locale.

II.3. Noțiuni introductive privind organizarea datelor

Organizarea datelor, în vederea prelucrării acestora, este un proces complex, dar de o deosebită importanță. De modul în care sunt structurate datele, depinde eficiența algoritmilor de prelucrare. Unele date sunt de tip simplu: data apare ca o entitate indivizibilă atât din punct de vedere a informației pe care o reprezintă, cât și în raport cu unitatea centrală de prelucrare. Alte date sunt descrise prin componente, cu același tip sau cu tipuri diferite. O colecție de date pe care s-a evidențiat un anumit mod de structurare și s-au stabilit procedeele de înregistrare/identificare a componentelor se va numi structură de date. Componentele unei structuri de date pot fi de tip elementar sau pot fi, la rândul lor, structuri. Identificarea unei componente se poate face prin poziția pe care o ocupă în structură sau prin nume.

Pentru o dată elementară trebuie specificate: un identificator, atrbute (domeniul de valori, modul de reprezentare în sistemul de calcul, precizia reprezentării) și valorile datei (pot fi enumerate sau indicate printr-o proprietate comună). Din punct de vedere al domeniului de valori asociat unei date se disting următoarele clase:

- date de tip integer (numere întregi);
- date de tip real sau float (cu elemente din mulțimea numerelor raționale);
- date de tip boolean (se referă la valorile de adevăr *true* (adevărat), *false* (fals));
- date de tip char (cu elemente ale mulțimilor ASCII sau Unicode);
- date de tip string (obținute prin concatenarea datelor de tip caracter);
- date de tip array (structură cu componente de același tip ce ocupă locații succesive din memoria sistemului de calcul, identificate prin poziție);
- date de tip record (structură cu componente oarecare, identificate prin nume).

Un mod special de organizare a datelor este întâlnit când avem de prelucrat liste. O listă liniară este o structură de date omogenă, secvențială, formată din elemente aparținând unei mulțimi date. O listă poate fi vidă (nu are nici un element) sau plină (nu mai există spațiu pentru stocarea unor componente suplimentare). Este foarte important să putem accesa un element al listei, să inserăm sau să ștergem un element etc.

Listele pot fi stocate în memoria unui sistem de calcul în două moduri: secvențial și înlănțuit. Modul secvențial presupune stocarea elementelor listei în locații succesive de memorie conform ordinii elementelor din listă și reținerea adresei primului element al listei (adresa de bază). Modul înlănțuit presupune că fiecare element al listei este înlocuit cu o celulă formată dintr-o parte de informație (corespunzătoare elementului listei) și o parte de legătura ce conține adresa celulei următorului element din listă. Se va retine adresa de bază a listei, iar ultima celulă va indica, în partea de legătura, o valoare specială (ce nu poate desemna o legătură).

Structura de date elementară adecvată reprezentării secvențiale a listelor este tabloul unidimensional.

Orice listă liniară are un început și un sfârșit pe care le numim bază, respectiv vârf. O listă liniară la care inserarea și extragerea elementelor se face prin vârful listei se numește stivă (stack). O listă liniară în care inserările se efectuează la baza listei, iar extragerile prin vârful listei se numește coadă (queue).

Listele liniare pot fi transformate în liste circulare considerând că legătura ultimului element indică adresa bazei listei.

II.4. Limbaj algoritmic

O altă modalitate de reprezentare a algoritmilor o constituie utilizarea limbajului algoritmic. Limbajul algoritmic folosește o scriere similară limbajelor de programare moderne. El permite atât descrierea instrucțiunilor algoritmului, cât și descrierea exactă a tipului datelor cu care lucrează algoritmul. Un algoritm descris folosind limbajul algoritmic este o succesiune finită de declarări și instrucțiuni. Declarările precizează tipul și organizarea datelor. Ele apar înaintea instrucțiunilor ce descriu pașii algoritmului. Din punct de vedere al scrierii instrucțiunilor, o instrucțiune poate ocupa mai multe rânduri sau pe un rând pot fi scrise mai multe instrucțiuni. Instrucțiunile vor fi separate, între ele, folosind caracterul ';'.

Cuvintele care identifică un tip de date sau o instrucțiune, numite în continuare cuvinte cheie, apar evidențiate pentru a fi deosebite de numele variabilelor. O declarare utilizează unul dintre cuvintele cheie integer, real, boolean, char, string, array, record, stack, queue, urmat de o listă de nume de variabile. Declarările sunt separate, între ele, folosind caracterul ';'. Variabilele prezente într-o listă au tipul și organizarea precizată prin cuvântul cheie respectiv.

O importantă deosebită o au declarările de subprograme. În rezolvarea multor probleme apare necesitatea executării repetate a acelorași calcule pentru date diferite. Subprogramele permit descrierea acestor calcule o singură dată. Subprogramul poate fi apelat ori de câte ori este necesară efectuarea acestor operații. Un subprogram este identificat printr-un nume și o listă de parametri. Subprogramele se împart în proceduri și funcții. Declararea unei proceduri constă în specificarea cuvântului procedure, a unui identificator al procedurii și a unei liste de declarări (între paranteze rotunde) ce indică informațiile ce fac obiectul transferului între apelant și apelat. Pentru declaraarea unei funcții se folosește cuvântul cheie function. Spre deosebire de proceduri, funcțiile întorc obligatoriu un rezultat. De aceea,

în declarații, declararea unei funcții începe cu specificarea mulțimii de valori ce corespunde rezultatului, a cuvântului function, a identificatorului funcției și a listei parametrilor (similar ca la o procedură).

Un algoritm este complet cunoscut dacă este descrisă și definiția subprogramelor folosite. Definiția unui subprogram presupune descrierea (prin instrucțiuni) modului în care se efectuează calculele și se transmit rezultatele. Mai multe detalii prezentăm în finalul acestei secțiuni.

Instrucțiunile limbajului algoritmic sunt următoarele:

1. Instrucțiunea de atribuire. Această instrucțiune are forma: $v := E$ (atribuire simplă) sau $(v_1, v_2, \dots, v_n) := (E_1, E_2, \dots, E_n)$ ce realizează simultan atribuirile $v_i := E_i$, pentru oricare $i = 1, 2, \dots, n$. Operația de atribuire este permisă când variabila v (variabilele v_1, v_2, \dots, v_n) din membru stâng și expresia E (expresiile E_1, E_2, \dots, E_n) sunt compatibile (se referă la aceeași clasă de obiecte). O expresie conține paranteze (optional), operanzi (inclusiv apeluri de funcții) și operatori adecvați.

2. Instrucțiuni de intrare/ieșire. Vom presupune că citirea datelor (de intrare) se face de la un mediu de intrare (de exemplu: tastatura sistemului de calcul), iar scrierea rezultatelor (de ieșire) se face la un mediu de ieșire (de exemplu: ecranul, imprimanta, plotterul etc.). Forma instrucțiunilor de intrare/ieșire este:

read v_1, v_2, \dots, v_n

write v_1, v_2, \dots, v_n

unde v_1, v_2, \dots, v_n sunt variabile de tip elementar.

3. Instrucțiunea repetitivă *While*. Această instrucțiune are forma: `while p do S`, unde p este un predicat, iar S este o secvență de instrucțiuni. Deoarece instrucțiunile sunt separate între ele, folosind ';' va trebui să delimităm secvența S . Pentru aceasta se utilizează construcția SEQ..END prezentată mai sus. Semnificația acestei instrucțiuni este aceeași ca pentru subschema logică *While(p; S)*.

4. Instrucțiunea *If_then_else*. Această instrucțiune are forma: `if p then S1 [elseS2]`, unde p este un predicat, iar $S1$ și $S2$ sunt secvențe de instrucțiuni. Dacă neîndeplinirea predicatorului p nu indică vreo acțiune, porțiunea *else S2* poate lipsi, fapt reprezentat prin includerea între paranteze drepte, exprimarea fiind echivalentă cu $IF0(p; S1)$. Atunci când atât $S1$, cât și $S2$ sunt acțiuni prevăzute, instrucțiunea este echivalentă cu subschema logică $IF(p; S1, S2)$.

5. Instrucțiunile *insert* și *extract*. Aceste instrucțiuni sunt necesare pentru lucrul cu liste. Acestea sunt o prelungire a instrucțiunii de atribuire. Dacă se specifică o listă L , atunci *insert i, L* (sau $L:=i$) exprimă introducerea elementului specificat prin i în lista L , iar instrucțiunea *extract i, L* (sau $i:=L$) specifică extragerea elementului curent din lista L și depunerea acestuia în i .

6. Instrucțiunea *apel* de procedură. Apelarea unei proceduri se face prin instrucțiunea *apel* de procedură care are una din formele:

identificator_procedura

sau

identificator_procedura(lista de argumente)

unde *identificator_procedura* specifică o procedură declarată, iar argumentele sunt expresii separate prin virgulă.

Se presupune că, atunci când se ajunge la un apel de procedură, se stabilește corespondența între argumente și parametri, și se execută toate instrucțiunile specificate în definiția procedurii. După ultima instrucțiune a procedurii se continuă cu instrucțiunea următoare apelului de procedură. Un apel de procedură este corect atunci când, între argumente și parametri există o concordanță ca număr, tip și mod de organizare. Convenim ca atunci când referim o variabilă (într-o procedură) de fapt, facem o referire la locația de memorie corespunzătoare argumentului respectiv. Spunem că se realizează un transfer prin referință. Sunt posibile și alte moduri de transfer, dar acestea nu sunt considerate momentan.

7. Instrucțiunea *return*. Această instrucțiune provoacă părăsirea corpului unui subprogram. În cazul în care cuvântul *return* este urmat de o expresie, valoarea expresiei este folosită ca valoare de return a subprogramului. Instrucțiunea *return* fără valoare de return este folosită pentru a părăsi execuția unei proceduri și a reveni în unitatea de program din care a avut loc apelul; și anume la instrucțiunea ce urmează imediat acestui apel.

8. Pentru a ușura descrierea algoritmilor admitem prezența, ca instrucțiuni ale limbajului algoritmic, a instrucțiunilor *Repeat* și *For*. Instrucțiunea *Repeat* este modelată de structura repetitivă REPEAT (p; S). Ea are forma: Repeat S until p; unde S este o secvență (eventual vidă) de instrucțiuni, iar p modelează o expresie logică.

Instrucțiunea *For* este modelată de structura iterativă FOR(p; a,b,c) și are forma simplificată

For v := e₁, e₂, e₃ do S

unde: S este o secvență de instrucțiuni, iar expresiile e₁, e₂ și e₃ au același domeniu de valori ca variabila v. În forma prezentată, instrucțiunea determină executarea secvenței S pentru v luând succesiv valorile e₁, e₁+e₃, e₁+2e₃, ..., fără a trece dincolo de e₂.

Formei de mai sus îi corespunde structura iterativă:

FOR((v-v₂) * v₃ ≤ 0; SEQ v := e₁; v₁ := e₂; v₃ := e₃ END, S, v := v + v₃).

II.5. Analiza complexității

Existența unui algoritm de rezolvare a unei probleme generează, imediat, întrebări ca: *Mai există un alt algoritm de rezolvare? Este algoritmul găsit "cel mai rapid"?* Acest paragraf introduce noțiunile fundamentale privind complexitatea algoritmilor și prezintă exemple simple de algoritmi pentru care se determină complexitatea.

Analiza complexității unui algoritm presupune determinarea resurselor de care acesta are nevoie pentru a produce datele de ieșire. Prin resursă înțelegem timpul de executare, dar uneori este necesar să analizăm și alte resurse precum: memoria internă, memoria externă etc. *Modelul mașinii pe care va fi executat algoritmul nu presupune existența operațiilor paralele; operațiile se execută secvențial.*

Timpul de executare al unui algoritm reprezintă numărul de operații primitive executate. Trebuie, pentru fiecare algoritm, să definim noțiunea de

operăie primitivă, independent de mașina secvențială pe care se va execuă algoritmul.

În analiza complexităii unui algoritm avem în vedere cazul cel mai defavorabil din mai multe motive:

1. Timpul de executare în cazul cel mai defavorabil oferă o limită superioară a timpului de executare (avem certitudinea că executarea algoritmului nu va dura mai mult).
2. Situația cea mai defavorabilă este întâlnită des.
3. Timpul mediu de executare este, uneori, apropiat de timpul de executare în cazul cel mai defavorabil, dar dificil de estimat.

Notația theta este utilizată pentru a specifica faptul că o funcție este mărginită (inferior și superior). Semnificația notației O este de limită superioară, în timp ce semnificația notației Omega este de limită inferioară.

Definiție. Fie A un algoritm, n dimensiunea datelor de intrare și $T(n)$ timpul de executare estimat pentru algoritmul A . Se spune că algoritmul A are comportare polinomială (apartine clasei P) dacă există $p > 0$, astfel încât $T(n) = O(n^p)$.

Definiție. O funcție care crește mai rapid decât funcția putere x^p , dar mai lent decât funcția exponențială a^x cu $a > 1$ se spune că este cu creștere exponențială moderată. Mai precis: f este cu creștere exponențială moderată dacă pentru oricare $p > 0$ avem $f(x) = \Omega(n^p)$ și oricare $M > 0$ avem $f(x) = o((1+M)^x)$.

Definiție. O funcție f are creștere exponențială dacă există $a > 1$ astfel încât $f(x) = \Omega(a^x)$ și există $b > 1$ astfel încât $f(x) = O(b^x)$.

Prințre funcțiile $n \rightarrow f(n)$, nemărginite, funcțiile ce cresc cel mai lent sunt, de exemplu, de forma $\log \log n$ sau $(\log \log n)^{1,02}$. Pentru $n = 1000000$, $\log \log n \sim 2,6$. Deci un algoritm a cărui complexitate este $\log \log n$ este preferabil unui algoritm (elaborat pentru rezolvarea aceleiași probleme) de complexitate $\log n$. Algoritmii de tip polinomial sunt mai lenți (creșterea funcției $T(n)$ este mai rapidă) decât algoritmii de tip logaritmic. Urmează apoi algoritmii moderați (cu $T(n)$ de forma $n^{\log n}$ etc.) și cei cu creștere exponențială (2^n , $n^3 3^n$ etc.). Algoritmii cei mai lenți sunt cei pentru care funcția $T(n)$ este foarte rapid crescătoare (de exemplu: $T(n) = n!$).

În informatică sunt interesanți numai algoritmii care conduc la un timp de calcul cel mult moderat. Dacă un algoritm necesită timp de calcul exponențial sau factorial, acesta va fi utilizat numai în cazuri excepționale. Tabelul următor ilustrează cele de mai sus.

n	10	100	1000	10000
n^2	100	10000	1000000	100000000
n^3	1000	1000000	1000000000	1000000000000000
$\log n$	1	2	3	4
$\log \log n$	0	0.30103	0.47712	0.60206

Propoziție. Fie n și m două numere întregi pozitive. Algoritmul lui Euclid pentru a determina $\text{cmmdc}(m, n)$ efectuează cel mult $[2\log_2 M] + 1$ operații de împărțire întreagă, unde $M = \max(m, n)$.

II.6. Elemente privind corectitudinea algoritmilor

A verifica corectitudinea unui algoritm înseamnă a verifica dacă algoritmul conduce într-un interval finit de timp la obținerea soluției corecte a problemei pentru care a fost elaborat. Vom vedea în capitolul 5 câteva metode de rezolvare a problemelor, deci de a elabora algoritmi. Metodele descrise în acest capitol se vor exemplifica pentru algoritmi simpli. Pentru aspecte suplimentare legate de corectitudinea algoritmilor se poate folosi lucrarea: *Tudor Bălănescu. Corectitudinea algoritmilor. Editura Tehnică, București, 1995*.

Notăție. Construcția $\{P\}A\{Q\}$, numită și formulă de corectitudine totală conține următoarele elemente:

P - comentariu care descrie proprietățile datelor de intrare (precondiția);

A - algoritmul (secvența de instrucțiuni) supus analizei;

Q - comentariu care descrie proprietățile datelor de ieșire (postcondiția).

Definiție. Un algoritm $\{P\}A\{Q\}$ este corect când propoziția următoare este validă:

Dacă

datele de intrare satisfac precondiția P

Atunci

1) executarea lui A se termină (într-un interval finit de timp) și

2) datele de ieșire satisfac postcondiția Q.

Folosind elementele fundamentale ale logicii matematice rezultă că următoarele observații sunt adevărate:

1. Algoritmul $\{\text{false}\}A\{Q\}$ este corect, oricare ar fi A și Q.
2. Algoritmul $\{P\}A\{\text{true}\}$ este corect dacă și numai dacă executarea lui A se termină, atunci când datele inițiale satisfac proprietatea P.
3. Dacă $\{P\}A\{Q\}$ și $\{R\}A\{Q\}$ sunt formule corecte, atunci $\{P \vee R\}A\{Q\}$ este formulă corectă.

Pentru a stabili corectitudinea algoritmilor complecsi se procedează la descompunerea acestora în elemente simple a căror corectitudine se analizează. În continuare vor fi prezentate reguli pentru a analiza corectitudinea unor astfel de algoritmi. Pentru o formalizarea avansată a acestor reguli, cititorul interesat poate parurge lucrarea: *C. A. R. Hoare et al. Laws of Programming. Comm. ACM. 30(8), 1987, 672-687*.

Regula compunerii secvențiale (CS):

Dacă A este de forma SEQ B; C END, atunci se verifica formula $\{P\}A\{Q\}$, revine la a verifica formulele $\{P\}B\{R\}$ și $\{R\}C\{Q\}$, unde R este un predicat asupra datelor intermediare.

Este evident că regula CS poate fi aplicată iterativ. Mai precis, dacă A este de forma SEQ A₁; A₂; ..., A_n END, atunci obținem regula compunerii secvențiale generale:

&t9;CSG: Dacă $\{P_0\} A_1 \{P_1\}, \{P_1\} A_2 \{P_2\}, \dots, \{P_{n-2}\} A_{n-1} \{P_{n-1}\}$ și $\{P_{n-1}\} A_n \{P_n\}$ sunt algoritmi corecți, atunci $\{P_0\} A \{P_n\}$ este algoritm corect.

Regula implicației (I):

Această regulă este utilă în cazul algoritmilor ce urmează a fi aplicați în condiții mai tari decât pentru cele care au fost deja verificate.

O proprietate P este mai tare decât proprietatea Q dacă este adevărată propoziția compusă: $P \rightarrow Q$.

Regula implicației are următoarea formă:

I: Dacă $\{P\} A \{Q\}$ este algoritm corect, $P_1 \rightarrow P$ și $Q \rightarrow Q_1$, atunci $\{P_1\} A \{Q_1\}$ este algoritm corect.

Regula instrucțiunii de atribuire (A):

Fie notațiile:

x	Variabilă simplă
e	Expresie;
Def(e)	Proprietatea satisfăcută de acele elemente pentru care evaluarea expresiei e este corectă (Exemplu: pentru integer array b(10), Def(b(i)) := (i=1, 2, ..., 10));
P(x)	Formulă în care apare variabila x;
P(x/e)	Formula obținută din P(x) prin substituirea variabilei simple x cu expresia e, ori de câte ori x este variabilă liberă în P, iar dacă e conține o variabilă y care apare legată în P, înainte de substituție variabila y se înlocuiește printr-o nouă variabilă (care nu mai apare în e).

Valoarea de adevăr a propoziției $P \rightarrow Q(x/e)$ nu se schimbă dacă în $Q(x/e)$ se efectuează substituția descrisă de atribuirea $x := e$. Regula atribuirii este deci:

A: Dacă $P \rightarrow (\text{Def}(e))$ și $Q(x/e)$ atunci algoritmul $\{P\} x := e \{Q\}$ este corect.

Regula instrucțiunii if (IF si IFR):

Dacă c este o expresie booleană și A și B sunt algoritmi, pentru cele două forme ale instrucțiunii if sunt valabile regulile de corectitudine:

IF: Dacă

$\{P \text{ and } c\} A \{Q\}, \{P \text{ and not } c\} B \{Q\}$ sunt corecte, iar $P \rightarrow \text{Def}(c)$ este adevărată
Atunci formula

$\{P\} \text{ if } c \text{ then } A \text{ else } B \{Q\}$ este corectă.

IFR: Dacă

$\{P \text{ and } c\} A \{Q\}$ este corectă, iar $P \text{ and (not } c\}) \rightarrow Q$ și $P \rightarrow \text{Def}(c)$ sunt adevărate

Atunci

$\{P\} \text{ if } c \text{ then } \{Q\}$ este formulă corectă.

Se poate observa că aplicarea regulilor instrucțiunii de decizie nu este în sine dificilă corectitudinea acestei instrucțiuni se reduce la corectitudinea instrucțiunilor componente.

Regula instrucțiunii while (W):

Regula instrucțiunii while trebuie să precizeze dacă nu apare fenomenul de ciclare, iar prelucrările sunt corecte (în paranteze rotunde apare descrierea formală).

Fie algoritmul $\{P\} A; \text{while } c \text{ do } S \{Q\}$. Presupunem că există o proprietate invariantă I (vezi mai jos) și o funcție de terminare t cu valori numere întregi care satisfac următoarele condiții:

- Când I este adevărată atunci expresia booleană c este bine definită (adică $I \rightarrow \text{Def}(c)$).
- Proprietatea I rezultă prin executarea secvenței A (adică, $\{P\} A \{I\}$ este algoritm corect).
- La terminarea instrucțiunii while, proprietatea finală Q poate fi dedusă (adică, $I \text{ and } (\text{not } C) \rightarrow Q$).
- I este proprietate invariantă la executarea unei iterații: dacă I este adevărată înainte de executarea secvenței S și expresia booleană c este adevărată, atunci executarea secvenței S se termină într-un interval finit de timp și I este adevărată la sfârșit (adică, $\{I \text{ and } c\} S \{I\}$ este algoritm corect).
- Dacă rezultatul evaluării expresiei c este *true* și proprietatea I este adevărată, atunci există cel puțin o iterație de efectuat (adică, $I \text{ and } c \rightarrow (t >= 1)$).
- Valoarea lui t descrește după executarea unei iterații (adică, $\{(I \text{ and } c) \text{ and } (t=a)\} S \{t < a\}$).

În aceste condiții, algoritmul considerat este corect.

III. LIMBAJE DE PROGRAMARE

III.1. Vocabularul și sintaxa limbajelor de programare

Vocabularul unui limbaj de programare este format din cele mai simple elemente cu semnificație lingvistică numite entități lexicale sau *tokens*. Elementele vocabularului sunt alcătuite din caractere Unicode (care constituie alfabetul limbajului). Standardul Unicode conține ca subset codul ASCII, dar reprezentarea internă a caracterelor Unicode folosește 16 biți. Cele mai utilizate simboluri sunt: literele mari și mici ale alfabetului englez, cifrele sistemului zecimal, diferite semne speciale.

Unitățile lexicale sunt separate, între ele, prin comentarii și spatii. Pentru aproape toate limbajele de programare se pot evidenția unități lexicale precum: cuvinte cheie, identificatori, literali, separatori și operatori. Cuvintele cheie sunt secvențe de caractere ASCII rezervate (nu pot avea altă semnificație) utilizate pentru definirea unităților sintactice fundamentale. Pentru exemplificare ne referim la limbajele de programare *Pascal* și *Java*:

1. *Cuvinte cheie Pascal*: absolute, and, array, begin, case, const, div, do, downto, else, end, external, file, for, forward, function, goto, if, implementation, in, inline, interface, interrupt, label, mod, nil, not, of, or, packed, procedure, program, record, repeat, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor.

2. *Cuvinte cheie Java*: abstract, boolean, break, byte, case, cast, catch, char, class, const, continue, default, do, double, else, extends, final, finally, float, for, future, generic, goto, if, implements, import, inner, instanceof, int, interface, long, native, new, null, operator, outer, package, private, protected, public, rest, return, short, static, super, switch, synchronized, this, throw, throws, transient, try, var, void, volatile, while, byvalue. Cuvintele cheie subliniate sunt prezente și în limbajul C alături de altele.

Identifierii sunt secvențe, teoretic nelimitate, de litere și cifre Unicode, începând cu o literă sau liniuță de subliniere (în limbajul C). Identifierii nu pot fi identici cu cuvintele rezervate.

Literalii ne permit introducerea valorilor pe care le pot lua tipurile de date primitive și tipul sir de caractere. Mai precis, literalii sunt constante întregi, flotante, booleene, caracter și, siruri de caractere.

Literalii întregi, în general, pot fi descriși prin reprezentări în una din bazele de numerație: 10, 16 sau 8. Lungimea reprezentării interne depinde de implementarea limbajului. De exemplu, în limbajul Pascal, un literal întreg, cu semn, este reprezentat pe 16 biți, descrierea sa în baza 16 fiind o secvență a simbolurilor asociate reprezentării numărului întreg în baza 16 având prefixul \$.

Literalii flotați reprezintă numere rationale. Ei sunt formați din următoarele elemente: partea întreagă, partea fracționară și exponent. Exponentul, dacă există, este introdus de litera E sau e urmată optional de un semn al exponentului.

Literalii booleeni sunt TRUE și FALSE, primul reprezentând valoarea booleană de adevăr, iar celălalt valoarea booleană de fals. Deși TRUE și FALSE nu sunt cuvinte rezervate, acestea nu pot fi folosite ca identifieri (în Pascal, Java).

Literalii caracter sunt folosiți pentru a desemna caracterele codului Unicode (sau ASCII, acolo unde este cazul). Descrierea unui literal caracter se fie folosind o literă, fie o secvență specială. Secvențele speciale (numite secvențe escape în C, C++ și Java) permit specificarea caracterelor fără reprezentare grafică precum și a unor caractere speciale. Caracterele ce au reprezentare grafică pot fi descrise între apostrofuri, ca în exemplele: 'P', 'A', '3', '!'. Secvențele speciale se descriu diferit de la un limbaj la altul. Vom exemplifica folosind limbajele Pascal și Java.

Secvențe speciale în limbajul Pascal: 1) Un întreg din domeniul 0, ..., 255 precedat de simbolul # desemnează un caracter dat prin codul său ASCII. 2) Un caracter imprimabil precedat de semnul ^ desemnează un "caracter de control" având codul ASCII în domeniul 0, ..., 31.

Secvențele *escape* în limbajul Java se descriu folosind apostrofuri, semnul \, litere și cifre. Vom exemplifica indicând câteva secvențe predefinite: '\b' (backspace = #8), '\n' (linefeed), '\r' (carriage return) etc.

Un literal sir de caractere este constituit din zero sau mai multe caractere intre delimitatori. Secvența de caractere ce formează sirul poate conține atât caractere imprimabile, cât și secvențe speciale. În limbajul Pascal se utilizează apostroful ca delimitator, iar în limbajul C(C++, Java) ghilimelele.

Separatorii sunt caractere ce indică sfârșitul unui *token* și începutul altuia. Aceștia participă și la construcția sintaxei limbajelor de programare. Ei nu trebuie confundați cu spatiile care, și acestea, separă unități lexicale distincte. Separatorii sunt necesari când unitățile lexicale diferite sunt scrise fără spații între ele. Cei mai întâlniți separatori sunt: () { } [] ; , . Exemple: x[10], f(x,y), carte.autor etc. Operatorii sunt simboluri grafice ce desemnează operațiile definite de un limbaj de programare. În unele limbi de programare este posibilă redefinirea operatorilor, același simbol fiind utilizat pentru operații diferite ce rezultă din contextul în care apar. Lista minimală a operatorilor aritmici include: +(adunare), -(scădere), /(împărțire), *(înmulțire). Mai sunt admise și operații precum: % (C, Java) sau mod (Pascal, Modula), div (împărțire întreagă în limbajul Pascal). Alți operatori sunt: operatori logici, operatori relaționali, operatori asupra sirurilor de caractere etc. Toți operatorii pot fi priviți și ca separatori.

O construcție aparte utilizată în programe pentru explicarea sau documentarea textului programului este comentariul. Comentariile sunt delimitate de textul programului folosind anumiți delimitatori. În limbajul Pascal, un comentariu este scris între acoladele { } sau între secvențele (*, *). Programele C++, Java pot conține comentarii pe o singură linie și încep cu //, sau pe mai multe linii și sunt cuprinse între /* și */.

Alte elemente lexicale ce pot fi prezente într-un program sunt etichetele și clauzele. Etichetele sunt siruri de cifre zecimale/hexazecimale sau identificatori folosite în legătura cu o instrucțiune de salt (*goto*) pentru marcarea unor instrucțiuni. Clauzele (numite și directive) sunt cuvinte cheie ce desemnează instrucțiuni cu efect în timpul compilării.

Prin sintaxa unui limbaj de programare se înțelege, în general, un ansamblu de reguli privind aggregarea unităților lexicale pentru a forma structuri mai complexe (declarații, instrucțiuni, module, programe etc.). Prezentarea acestor reguli se poate folosind limbajul natural sau mecanisme formalizate. Descrierea sintaxei în limbaj natural poate conduce la neclarități sau specificații incomplete. Cu ajutorul mecanismelor formale sintaxa unui limbaj este complet specificată. Cea mai folosită notație este cunoscută sub numele de notație BNF (Backus-Naum-Form) și a fost folosită pentru prima dată, în anul 1959, la specificarea sintaxei limbajului Algol-60. Această notație are aceeași putere generativă cu gramaticile independente de context introduse de N. Chomsky. Totuși, limbajele de programare nu sunt independente de context, ci numai porțiuni ale acestora pot fi modelate cu ajutorul limbajelor independente de context. Pentru a putea specifica un întreg limbaj de programare se poate folosi notația BNF extinsă. În prezentarea din acest capitol vom utiliza opt metasimboluri: ::= <> { } [] | pentru a defini unitățile sintactice ale limbajului Pascal. Metasimbolurile < și > sunt folosite pentru delimitarea numelui unei unități sintactice. Presupunem, de asemenea, existența unei operații de concatenare pe mulțimea unităților sintactice.

Metasimbolul ::= apare după numele unei unități sintactice și are semnificația "se definește prin". Metasimbolul | este utilizat pentru a delimita mai multe variante de definire ale unei unități sintactice, aceasta fiind obținuta prin reuniunea variantelor. Metasimbolurile { și } indică repetarea posibilă (de zero sau mai multe ori) a simbolurilor pe care le delimitizează. Pentru a desemna prezenta optională a unor simboluri se utilizează, ca delimitatori, metasimbolurile [și]. Vom admite, pentru prescurtare metasimbolul ... care indică continuarea unui sir de valori conform contextului în care apare. Iată câteva exemple:

1. <literă> ::= A ... Z descrie mulțimea literelor mari;
2. <cifră> ::= 0 ... 9 descrie mulțimea cifrelor zecimale;
3. <identificator> ::= <literă> { <literă> | <cifră>} descrie modul de formare a identificatorilor: un sir de litere și/sau cifre, primul semn fiind o literă.
4. <secvența cifre> ::= <cifră> { <cifră>} descrie modul de formare a unei secvențe de cifre;
5. <întreg fără semn> ::= <secvența cifre> definește un număr întreg fără semn;
6. <semn> ::= + | -
7. <întreg cu semn> ::= [<semn><întreg fără semn> spune că un întreg cu semn este un întreg fără semn precedat de un semn: + sau -.

Prin diagrame sintactice se realizează o reprezentare grafică a modului de agregare a unităților sintactice. În cele ce urmează vom prefera limbajul natural (în anumite cazuri) și notația BNF extinsă (în alte cazuri), cititorul interesat asupra diagramelor sintactice poate consulta, de exemplu: *N. Wirth: Systematic Programming: An introduction, Prentice Hall, 1972*.

III.2. Tipuri de date. Constante. Variabile. Expresii

Un tip de date este o structură compusă din: 1) o mulțime X de valori numite date și 2) o mulțime de legi de compoziție pe X (operații ce se pot efectua cu valori din X). O dată are un singur tip (apartine unei singure mulțimi). Există limbaje de programare puternic tipizate (în sensul verificării cu regularitate a apartenenței unei date la mulțimea de valori a tipului său, încă din faza de compilare). Astfel de limbaje de programare sunt: Pascal, Modula, Ada etc.

Tipurile de date sunt standard sau definite de utilizator. Tipurile definite de utilizator se introduc prin intermediul unei definiții folosind un cuvânt cheie precum *type* (în Pascal), *typedef* (în C) sau *class* (în limbajele C++ și Java). De asemenea, se vor utiliza diverse cuvinte cheie pentru a specifica structura tipului. Dacă pentru o anumită structură a unui tip nu este stabilit un identificator, spunem că avem de-a face cu un tip anonim.

Valorile unui tip de date (elementele mulțimii X) sunt referite fie prin variabile, fie prin constante (literali sau constante simbolice). O locație de memorie care poate stoca o valoare a unui anumit tip de date se numește, prin abuz de limbaj, variabilă. Orice variabilă trebuie să fie declarată pentru a putea fi folosită. O declarație conține un tip de valori - ce indică: ce se stochează, cum se stochează și în ce operații intervin valorile stocate - și un identificator pentru a ne referi la

variabila ca obiectul declarației. Practic o variabilă este un obiect caracterizat de tip, adresă și valoare, pentru care atributul valoare poate fi modificat.

Operațiile cu elemente ale unui tip sunt fie predefinite, fie sunt introduse prin declarații *function* sau *procedure* (în Pascal) sau *operator* (în C++). Agregarea variabilelor, constantelor și a operatorilor conduce la construcții numite expresii. Expresiile sunt evaluate în cursul executării unui program. Rezultatul unei expresii depinde de valorile variabilelor în momentul evaluării.

Tipurile de date întâlnite în limbajele de programare actuale sunt clasificate în: tipuri de date simple; tipuri de date structurate, tipuri referință (pointer), tipuri procedurale. În limbajele C, C++ și Java există tipul *void*. Această mulțime notată prin *void* înseamnă fie mulțimea vidă, fie o mulțime neprecizată.

Tipurile de date simple numite și tipuri primitive (sau tipuri standard) se referă la mulțimi de elemente precum: numere întregi, numere raționale, valori de adevăr (logice sau booleene), caractere, valori aparținând unei enumerări sau unui interval (subdomeniu). O parte dintre tipurile simple sunt tipuri ordinale, adică tipuri caracterizate printr-o mulțime finită de valori, pe care este definită o ordine liniară și, prin urmare, pentru orice element al unei asemenea mulțimi se stabilește numărul de ordine *ord()*, elementul predecesor *pred()* și cel succesor *succ()*. Tipurile ordinale sunt cele care se referă la mulțimi precum: mulțimea numerelor întregi, mulțimea valorilor de adevăr, mulțimea caracterelor, mulțimea valorilor unei enumerări, mulțimea valorilor dintr-un subdomeniu al uneia dintre mulțimile anterioare. Tipurile raționale (simplă precizie, dublă precizie, precizie extinsă etc.) nu sunt considerate tipuri ordinale, deși sunt tot mulțimi finite de elemente. Trebuie observat că metoda de reprezentare în memoria calculatorului a numerelor raționale ar permite considerarea unei ordini liniare și, elementele unei astfel de mulțimi ar avea un număr de ordine.

Operatorii sunt simboluri care specifică operații efectuate asupra unor variabile sau constante denumite operanzi. Combinățiile valide de operanzi și operatorii reprezintă expresii.

Limbajele de programare oferă o multitudine de operatori: operator de atribuire (simbolizat prin := sau =), operatori aritmetici unari (utilizarea semnului, incrementare, decrementare), operatori aritmetici binari (adunare, scădere, înmulțire, împărțire, obținerea câtului și restului împărțirii a două numere întregi), operatori logici (și, sau, negare), operatori relaționali (egal, diferit, mai mic, mai mic sau egal, mai mare, mai mare sau egal, aparține), operatori la nivel de bit (și, sau, negare, sau exclusiv, deplasare stânga, deplasare dreapta), operatori combinați (în limbajele C, C++ și Java), operatori asupra mulțimilor (reuniune, intersecție, diferență), operatori asupra sirurilor de caractere (concatenare) precum și alți operatori.

Evaluarea expresiilor trebuie să tină seama de poziția parantezelor și de proprietățile operatorilor (precedență, asociativitate, conversii implicate în cazul tipurilor compatibile, conversii explicite). Precedenta stabilește ordinea de evaluare a operațiilor în cazul expresiilor care conțin mai mulți operatori diferiți. Dacă într-o expresie se întâlnesc operații cu aceeași precedență, atunci ordinea de evaluare este dată de tipul asociativității (de la stânga la dreapta sau de la dreapta la stânga).

Când într-o expresie apar operații cu operanzi de tipuri diferite, înainte de efectuarea operației are loc un proces de conversie implicită (când nu se semnalează explicit și nu apare fenomenul de incompatibilitate) prin care tipul cu cardinalul mai mic este promovat către tipul mai bogat (se presupune aici că natura elementelor celor două tipuri este aceeași).

III.3. Programare în Turbo Pascal

Limbajul Pascal a fost definit de profesorul Niklaus Wirth (Universitatea Tehnică din Zürich, Elveția) în anul 1971. Există foarte multe implementări ale acestui limbaj, cea mai populară aparține firmei Borland International și este destinată calculatoarelor compatibile IBM-PC.

Structura unui program Turbo Pascal conține o parte declarativă și o parte principală. Partea declarativă conține o descriere a datelor ce vor fi prelucrate pe calculator, cât și o descriere a operațiilor ce se pot efectua asupra datelor. Partea principală (numită și corpul programului) descrie secvența de instrucțiuni ce se execută în momentul lansării programului pentru executare. Acțiunile (reprezentate de instrucțiuni) asupra datelor efective sunt descrise în partea principală. Partea declarativă indică un antet, o linie *uses*, una sau mai multe declarații (etichete (*label*), tipuri de date (*type*), constante (*const*) și variabile (*var*)) precum și una sau mai multe declarații și definiții de subprograme.

III.3.1. Instrucțiuni Pascal

Instrucțiunile limbajului Turbo Pascal sunt grupate în două clase: instrucțiuni simple și instrucțiuni structurate. Acțiunile simple sunt: instrucțiunea de atribuire, instrucțiunea apel de procedură, instrucțiunea de transfer necondiționat (*goto*) și instrucțiunea vidă (cu efect nul). Acțiunile structurate sunt reprezentate de: instrucțiunea compusă, instrucțiunile iterative (*for*, *while*, *repeat*), instrucțiuni condiționale (*if*, *case*) și instrucțiunea *with* (în contextul prelucrării datelor de tip record).

Din punct de vedere sintactic, o instrucțiune este alcătuită dintr-o etichetă (optională) pentru a putea fi referată de alte instrucțiuni urmată de instrucțiunea propriu-zisă (care descrie acțiunea realizată în momentul executării sale).

Instrucțiunea de atribuire, în notația BNF, are forma descrisă prin:

<instrucțiune de atribuire> ::= <variabilă> := <expresie>

unde <variabilă> și rezultatul dat de <expresie> trebuie să fie de tipuri identice, sau tipul uneia să fie un subdomeniu al celeilalte, sau ambele trebuie să fie subdomenii ale aceluiași tip. Se acceptă, ca o excepție, cazul în care <variabilă> are tipul real, iar <expresie> conduce la un rezultat de tip întreg.

Instrucțiunea apel de procedură se inserează în program în locul în care se dorește executarea instrucțiunilor specificate de o declarație de procedură asupra unor date efective transmise procedurii în locul de apel. Instrucțiunea write('Apel!'); apelează procedura write pentru sirul 'Apel !'. O procedură cu antetul procedure PROC(a, b, c:real; var s:real)

se poate apela în mai multe moduri. Următoarele apeluri sunt corecte:

PROC(3.0, 4.0, 5.0, S); PROC(L1,L2,L3,ARIA); PROC(2.0, Y, Z, A);

Apelul PROC(X,Y,Z,45.0); nu este un apel corect deoarece al patrulea parametru nu poate fi o constantă.

Instrucțiunea de transfer necondiționat, transferă controlul execuției programului în alt loc din textul sursă implementând mecanismul "Mergi la pasul ". Instrucțiunea către care se realizează transferul trebuie să fie una etichetată

Există anumite restricții privind utilizarea instrucțiunii *goto* (se transferă controlul în același bloc de instrucțiuni redat prin begin ... end sau repeat ... until; transferul către o instrucțiune din interiorul unei instrucțiuni compuse este permis, dar nu este recomandabil).

Instrucțiunea de efect nul (instrucțiunea vidă), nu are efect asupra variabilelor programului (nu schimbă starea programului). Ea nu este redată printre-un cuvânt cheie, dar deoarece instrucțiunile Pascal sunt separate prin delimitatorul ";", prezenta acesteia este marcată de apariția acestui delimitator. În exemplul: begin i:=i+1; ; write(i); end instrucțiunea cu efect nul apare între cei doi delimitatori ";" și între ";" și "end".

Instrucțiunea compusă. Când într-o instrucțiune este necesară specificarea unei alte instrucțiuni, se poate utiliza instrucțiunea compusă indicată prin secvența begin ... end. Instrucțiunile dintre *begin* și *end* se execută în ordinea în care apar. În secvența de mai jos,

```
i := 1;  
while <= n do  
    begin read(x); s := s + x; i := i + 1 end;
```

pentru fiecare *i* pentru care este adevărată condiția $i \leq N$ se vor executa instrucțiunile dintre *begin* și *end*.

Instrucțiuni iterative (repetitive). De foarte multe ori, un algoritm exprimă execuția de zero sau mai multe ori a unui grup de operații. Pentru specificarea acestui aspect se utilizează iterarea (ciclul sau bucla) concretizată într-o instrucțiune iterativă. Conform teoremei fundamentale a programării structurate, instrucțiunea repetitivă cu test inițial (*while*) este suficientă pentru exprimarea oricărui algoritm, deci și a oricărui program pentru calculator. Totuși, pentru uzul programatorilor, limbajele de programare includ și alte construcții. Instrucțiunile repetitive Pascal sunt: instrucțiunea *while*, instrucțiunea *repeat* (ciclul cu test final) și instrucțiunea *for* (ciclul cu contor).

Instrucțiunea while are forma While C do S; unde C este o expresie logică, iar S este o instrucțiune. Dacă S exprimă mai multe operații, atunci reprezintă o instrucțiune compusă. Instrucțiunea S se execută cât timp expresia logică C este adevărată. Dacă expresia logică C este falsă execuția continuă cu următoarea instrucțiune (imediat după *while*). Dacă de la început expresia C are valoarea false, atunci instrucțiunea S nu se execută. Trebuie observat că blocul S trebuie să asigure o schimbare, în timp, a valorii de adevăr a condiției C; altfel se obține o buclă infinită. Astfel, execuția secvenței: i:=1; While i<10 do read(x); va continua indefinit. Pentru a asigura terminarea ciclului trebuie să modificăm valoarea variabilei *i*. Secvența corectă este:

```
i:=1;  
while < 10 do begin read(x); i:=i+1 end;
```

Alte construcții interesante sunt:

While true do I; (o buclă infinită) și

While false do I; (instrucțiu de efect nul - secvența I nu se execută niciodată).

Instrucțiu de repeat are forma repeat S until C; unde S este o secvență de instrucționi (nu neapărat incluse într-o instrucțiu compusă), iar C este o expresie logică. Secvența S se execută cel puțin o dată, execuția ei continuând până când condiția C este îndeplinită (valoarea de adevăr a expresiei C este true). Practic, în loc de

```
repeat S until C;
```

putem scrie

```
S; while (not C) do S;
```

Instrucțiu de for are una din formele: for v:=i to f do S sau for v:=i downto f do S. Variabila v, numită contor, trebuie să fie de tip ordinal (pe mulțimea de definiție trebuie să aibă sens funcțiile succ(esor) și pred(cesor)), iar i și f trebuie să fie expresii compatibile cu tipul variabilei v. Forma for v:=i to f do S este echivalentă cu secvența:

```
v:=i; while v<= f do begin S; v := succ(v) end;
```

iar forma for v:=i downto f do S este echivalentă cu secvența:

```
v:=i; while v>=f do begin S; v:=pred(v) end;
```

Valoarea finală a variabilei v se consideră a fi nedefinită după terminarea normală a instrucționii *for*.

Instrucționi conditionale (instrucționi de decizie). O instrucțiu condițională selectează o singură instrucțiu dintre mai multe alternative posibile. Limbajul Pascal, alături de instrucțiu *if*, permite și utilizarea unei instrucționi de selecție multiplă: instrucțiu *case*.

Instrucțiu de if are una din forma: if C then S1 sau forma if C then S1 else S2. Pentru oricare dintre forme, inițial se evaluează expresia logică C. Dacă valoarea de adevăr a acestei condiții este *true* se va executa secvența de pe ramura *then* (S1), altfel în primul caz se execută instrucțiu de efect nul, iar în al doilea caz se execută secvența de pe ramura *else* (S2).

Instrucțiu de case utilizează o expresie numită selector și o listă de instrucționi, fiecare element din listă fiind precedat de o constantă cu un subdomeniu de valori din mulțimea de bază a selectorului. Conform notației BNF, forma instrucționii *case* este:

```
<instrucțiu de case> ::= case <selector> of  
[<element case> {; <element case>}] [else <instrucțiu>] [;] end  
<element case> ::= <constantă> {<constantă>}  
<constantă1> .. <constantă2>: <instrucțiu>
```

Expresia ce definește selectorul are valori ordonale cuprinse între -32768 și 32767.

Instrucțiu de with are forma generală:

with <variabilă record>{ ,<variabilă record>} do <instrucțiune>
 unde: lista dintre cuvintele rezervate *with* și *do* conține identificatorii variabilelor de tip înregistrare (record) cărora li se aplică instrucțiunea ce apare după cuvântul *do*. Această instrucțiune simplifică modul de referire la componentele înregistrărilor. În <instrucțiune> se folosesc numai identificatorii câmpurilor înregistrărilor specificate în lista dintre *with* și *do*.

III.3.2. Subprograme Pascal

Proceduri Pascal. Elementele introduse anterior: tip de date, variabilă, expresie, instrucțiune permit codificarea riguroasă a prelucrărilor de date sub formă de programe. Noțiunile de expresie și instrucțiune pot fi generalizate prin intermediul subprogramelor (funcții și proceduri). Se poate introduce tipul subprogram (numit și tip procedural).

Declarația unui subprogram Pascal asociază un identificator unui bloc de instrucțiuni precedat, eventual, de o secvență declarativă. Identificatorul subprogramului poate fi urmat de o listă de parametrii formali. Identificatorul unei proceduri (urmat, eventual, de o listă a parametrilor efectivi) poate fi folosit prin intermediul instrucțiunii apel de procedură. Identificatorul unei funcții urmat, eventual, de o listă a argumentelor efective, poate fi folosit în expresii sau în orice loc unde poate fi prezentă o expresie.

Declararea unei proceduri Pascal se realizează conform următoarelor reguli:

```
<procedură> ::= <antet_procedură>;<corp>
<antet_procedură> ::= procedure <identificator> [ (<listă_parametrii>)];
<corp> ::= [ interrupt ; | near ; | far ;] (bloc | forward; | external;| directivă INLINE | directivă ASM )
```

Instrucțiunile ce se execută la activarea unei proceduri (în general, a unui subprogram) sunt descrise în blocul subprogramului, în liniile INLINE sau în blocul ASM.

Funcții Pascal. Declararea unei funcții Pascal se supune regulilor următoare:

```
<funcție> ::= <antet>;<corp>
<antet> ::= = function <identificator> [ (listă_parametrii_formali) ] :<tip_rezultat>;
<corp> ::= [ near ; | far ;] (bloc | forward ;| external ;| directivă INLINE | directivă ASM)
```

Tipul rezultatului returnat este fie un nume de tip simplu, fie string.

Un subprogram *function* este activat printr-un apel de funcție. Apelul de funcție constă din identificatorul funcției urmat, eventual, de lista argumentelor funcției, între paranteze. Un apel de funcție apare ca un operand într-o expresie. La evaluarea expresiei, se apelează și se execută subprogramul *function*, iar valoarea operandului devine valoarea returnată de funcție. Zona de instrucțiuni a corpului specifică instrucțiunile ce urmează a-tilde; a fi executate la apelul funcției. Acest bloc trebuie să conțină cel puțin o instrucțiune de atribuire care asociază o valoare

identificatorului funcției. Rezultatul întors este ultima valoare asociată. Când o astfel de instrucțiune lipsește, valoarea returnată de funcție nu este precizată.

Comunicarea între unități de program. Comunicarea între programul principal și subprograme, precum și comunicarea între subprograme se poate face prin entități globale și prin parametri.

Comunicarea prin intermediul parametrilor permite tratarea subprogramelor ca pe niște blocuri specializate având un anumit număr de linii de intrare și un anumit număr de linii de ieșire. Această comunicarea este posibilă prin specificarea listei parametrilor la declararea și definirea unui subprogram. La execuția apelului subprogramului valorile parametrilor de apel sunt substituite parametrilor formali (blocul primește semnale concrete) astfel că acțiunile instrucțiunilor subprogramului au loc asupra acestor valori. Există trei categorii de parametri formali: parametri valoare, parametri variabilă, parametri fără tip.

Din punct de vedere sintactic, declararea parametrilor formali urmează regulile:

```
<listă_parametri_formali> ::= (<declaratie_parametru> { ; <declaratie_parametru> } )  
<declaratie_parametru> ::= <listă_identificatori> : <tip> |  
var <listă_identificatori> : <tip> |  
var <listă_identificatori>  
<tip> ::= <tip_simplu> | string | file
```

Un grup de identificatori neprecedați de var, dar urmăți de un tip reprezintă *parametri valoare*. Un parametru valoare, pentru subprogram, se comportă ca o variabilă locală, cu excepția faptului că la execuție va fi inițializat cu valoarea parametrului actual corespunzător. Orice schimbare asupra unui parametru valoare nu va afecta valoarea parametrului actual. Un parametru actual corespunzător unui parametru valoare, într-un apel de subprogram, trebuie să fie o expresie, nu de tip *file* și nici de tip structurat altul decât *string*. Parametrul actual și cel formal trebuie să aibă tipuri compatibile.

Un grup de parametri precedați de var și urmăți de un identificator de tip reprezintă parametri variabilă. Un parametru variabilă este utilizat atunci când subprogramul trebuie să furnizeze valori apelantului. Parametrul actual, într-un apel de subprogram, trebuie să fie declarat ca variabilă în unitatea apelantă sau mai sus, când este vorba de un identificator global. Parametrul formal de tip variabilă, în timpul unui apel, reprezintă chiar parametrul actual, deci orice modificare asupra parametrului formal se va reflecta asupra parametrului actual. Parametrul formal și parametrul actual corespunzător trebuie să fie de același tip. Fișierele pot fi transmise numai ca parametrii variabilă.

Un grup de parametri precedați de var, dar neurmăți de tip reprezintă parametrii fără tip. Cu ajutorul acestora se pot realiza prelucrări generice. Parametrul actual corespunzător unui parametru fără tip trebuie să fie declarat în unitatea apelantă sau mai sus ca variabilă. Utilizarea unui parametru fără tip, în subprogram, presupune o conversie de tip pentru a suporta corespondența cu parametrul actual.

Tipuri procedurale. Extinzând limbajul Pascal standard, Borland Pascal permite ca subprogramele să fie tratate ca elemente ale unor multimi de subprograme, deci se pot defini tipuri de date asociate subprogramelor. Un astfel de tip de date se numește tip procedural.

Odată definit un tip procedural, putem declara variabile de acest tip - numite variabile procedurale, iar aceste variabile pot fi folosite în atribuiră, cu operatorul `=`, cu operatorul `<>` și ca parametri în apelurile de subprograme.

Declarația unui tip procedural include parametri, iar pentru funcții, și tipul rezultatului. Regulile avute în vedere la o astfel de declarare sunt descrise prin:

```
<tip_procedural> ::= <tip_procedure> | <tip_function>
<tip_procedure> ::= procedure;
procedure (<listă_parametri_formal>);
<tip_function> :=
function : <tip_rezultat>;|
function (<listă_parametri_formali>):<tip_rezultat>;
```

Identifierii parametrilor în `<listă_parametri_formali>` sunt pur decorativi, ei nu au o semnificație deosebită în partea declarativă. Exemple:

```
type proc = procedure;
proc2int = procedure(var a, b:integer);
procstr = procedure(var s:string);
funcmat = function(x: real):real;
funcmat2 = function(x, y: real):real;
Eqfunc = function (a, b: real; f:funcmat): real;
```

Nu putem scrie subprograme de tip funcție care să returneze variabile procedurale. Funcțiile pot returna valori din următoarele categorii: string, real și variantele, integer și variantele, char, boolean, pointer (referință) și enumerare (definit de utilizator).

O variabilă procedurală se declară ca orice altă variabilă.

III.4. Programare în C

Limbajul C a fost creat la AT & T Bell Laboratories în anul 1972 de Dennis Ritchie. Versiunea standard a limbajului C până în anul 1988 a fost cea furnizată odată cu sistemul de operare UNIX și descrisă în [14]. În anul 1983 a început redactarea standardului ANSI pentru limbajul C. Standardul ANSI a fost finalizat în anul 1990.

III.4.1. Structura programelor C

În limbajul C programul este o colecție de module distincte numite funcții, organizate în una sau mai multe unități de translatare. Fiecare unitate de translatare poate fi compilată (analizată lexical și sintactic) separat. O unitate de translatare trebuie să conțină cel puțin o declarație sau o definiție de funcție. Ea constă din fișierul sursă împreună cu oricare fișier antet și fișiere sursă incluse prin directiva

#include. O unitate de translatare, prin compilare, conduce la un fișier obiect (.obj) relocabil.

Directivele precedate de delimitatorul # se numesc directive preprocesor, acestea specificând operații anterioare procesului de compilare ce sunt efectuate de o componentă a mediului de programare numită preprocesor.

O declarație C specifică atributelor unui identificator sau mulțime de identificatori. Regulile sintactice ce stau la baza scrierii declarațiilor sunt redate prin:

```
<declarație> ::= <specificator declarație>
[ <lista declaratori de inițializare> ] ;
<specificator declarație> ::= 
<clasa de memorare> [ <specificator declarație> ] |
<specificator tip> [ <specificator declarație> ] |
<calificator tip> [ <specificator declarație> ]
<lista declaratori de inițializate> ::= <declarator inițializare> |
<lista declaratori inițializare>, <declarator inițializare>
<declarator inițializare> ::= <declarator> |
<declarator> = <inițializare>
```

A face o declarație nu presupune și alocarea memoriei pentru identificatorul declarat. Există situații când alocarea se realizează în altă unitate de translatare (cazul datelor externe).

Declarația unui identificator asociază numelui în mod explicit sau implicit o serie de atrbute din mulțimea următoare:

- *Clasa de memorare* - localizează zona de memorie în care este plasat elementul declarat (zona de date, un registru al procesorului, stiva mediului de programare, zona de alocare dinamică) și delimită durata alocării (întreg timpul de executare a programului, executarea unei funcții sau a unui bloc etc.).
- *Domeniul numelui* - reprezintă porțiunea din program în care poate fi utilizat identificatorul pentru accesarea informației asociate și este determinat de poziția declarației.
- *Durata de stocare* - reprezintă perioada cât elementul asociat există efectiv în memorie.
- *Legătura* - indică modul de asociere a unui identificator cu un anumit obiect sau funcție, în procesul de editare a legăturilor.
- *Tipul datei* (standard sau definit de utilizator) - descrie informația conținută de elementul definit de identificator.

Clasa de memorare este specificată prin unul dintre cuvintele cheie: **typedef**, **extern**, **static**, **auto**, **register**. Declarația **auto** se poate utiliza pentru variabile temporare - alocate folosind stiva, cu domeniul local. Variabilele declarate în interiorul unui bloc sunt implicit locale, deci **auto** este rar utilizat. În limbajul C clasic, o declarație **register** reprezintă un apel la compilator pentru a stoca o variabilă **int** sau **char** într-un registru al procesorului pentru a crește viteza de executare. Versiunile actuale permit specificarea **register** pentru orice tip,

semnificația apelului fiind de optimizare a timpului de acces. Specificatorul **typedef** indică faptul că nu se declară o variabilă sau funcție de un anumit tip, ci se asociază un nume tipului de date. Sintaxa este:

typedef <definiție tip> <identificator>;

Specificatorul **static** poate să apară în declarații locale de variabile pentru a indica durata statică sau în declarații globale de funcții și de variabile pentru a indica legătura internă. Specificatorul **extern** este utilizat pentru declarații de funcții sau variabile locale sau globale pentru a indica legătura externă și durata statică.

În C (precum și în Pascal), declarația unei variabile trebuie să preceadă orice referire la ei. Ea poate apărea în exteriorul oricărei funcții, în lista de parametri formali ai unei funcții sau la începutul unui bloc. Domeniul numelui este regiunea dintr-un program C în care identificatorul este "vizibil". Poziția declarației determină următoarele domenii:

- *Domeniul bloc* - caracterizează identificatorii locali (identificatorii declarați în interiorul unui bloc și au domeniul cuprins între declarație și sfârșitul blocului; parametrii formali din definiția unei funcții au ca domeniu blocul funcției).
- *Domeniul fișier* - caracterizează identificatorii declarați în exteriorul oricărei funcții - numiți identificatori globali - și care au domeniul cuprins între declarație și sfârșitul fișierului.
- *Domeniul funcție* - aplicabil pentru etichetele instrucțiunilor și este blocul funcției.
- *Domeniul prototip* - definit pentru identificatorii specificați în lista de parametrii din prototipul unei funcții - și care au domeniul limitat la acel prototip.

Partea din domeniu în care informația asociată este accesibila se numește zona de vizibilitate. O declarație a unui identificator este vizibila în tot domeniul sau mai puțin blocurile sau funcțiile în care identificatorul este redeclarat. Pentru identificatorii globali se poate repeta declarația, dar inițializarea trebuie să se facă o singură dată.

Un identificator declarat în diferite domenii, de mai multe ori, sau redeclarat în același domeniu se poate referi la același obiect sau funcție prin procesul numit legare. Legarea poate fi internă, externă sau unică. Dacă un identificator are domeniul fișier și clasa de memorare static, el se supune legării interne. Dacă un identificator are clasa de memorare extern, el se supune aceluiași tip de legare precum orice declarație vizibila a identificatorului cu domeniu fișier; dacă nu există declarații vizibile cu domeniu fișier, se supune implicit legării externe. Pentru identificatorii cu legătura externă sunt permise mai multe declarații de referință, dar trebuie să existe o singură definiție. Funcțiile au implicit legătura externă și durata statică.

Specificatorii de tip indică modul de alocare asociat unei variabile sau tipul rezultatului unei funcții. În C, există următoarele categorii de tipuri: tipuri de funcții, tipuri de variabile și tipul **void**. Variabilele pot fi de tip scalar, de tip structurat sau de tip uniune. Tipurile scalare sunt tipuri aritmetice și tipul **pointer**. Tipurile structurate cuprind tablourile și înregistrările (numite în C, structuri). În

categoria tipurilor aritmetice intră mulțimile de elemente specificate prin cuvintele cheie: **char**, **int**, **float**, **double**; extinse cu ajutorul modificadorilor de tip: **signed**, **unsigned**, **short**, **long**. Tot tip aritmetic este considerat a fi și tipul obținut prin enumerare.

Tipul **void** indică absența oricărei valori și este utilizat în următoarele situații: declarația unei funcții fără parametri sau fără rezultat, tipul pointer generic și conversii de tip pentru pointeri.

Literalii sunt și ei afectați de existența modificadorilor de tip prin indicarea unui sufix (U, u, L, l, f, F). Efectul sufixului asociat unui literal întreg este ilustrat prin situațiile: U sau u - **unsigned int** sau **unsigned long int** (în funcție de valoare); L sau l - **long int** sau **unsigned long int** (în funcție de valoare); UL, ul, UL, uL - **unsigned long int**. Un literal de tip număr zecimal, este automat de tip **double**; dacă se utilizează sufixul F sau f va fi considerat de tip **float**, iar dacă se utilizează sufixul L sau l, va fi considerat de tip **long double**.

Tabloul este o listă de elemente de același tip plasate succesiv într-o zona contiguă de memorie. Nu există limită pentru numărul dimensiunilor tabloului.

Structura este o colecție de date eterogene (coresponde tipului **record** din limbajul Pascal). O declarație de structura precizează identificatorii și tipurile elementelor componente și constituie o definiție a unui tip de date nou. Acestui tip îl se poate asocia un nume. În cazul general, sintaxa declarației unei structuri este:

```
<declarație structura> ::=  
struct <id _tip> {  
<tip _camp _1> <id _camp _1>;  
<tip _camp _2> <id _camp _2>;  
...  
<tip _camp _i> <id _camp _i>;  
...  
<tip _camp _n> <id _camp _n>;  
} <lista identificatori de tip struct>;
```

în care:

- **struct** - este cuvânt cheie pentru construirea unui tip înregistrare;
- <**id _tip**> - este un identificator ce desemnează numele tipului structură ce este declarat;
- <**tip_camp_i**> - tipul câmpului i;
- <**id_camp_i**> - identificatorul câmpului i (câmpurile structurii se mai numesc și membrii structurii);
- <**lista identificatori de tip struct**> - lista identificatorilor declarați.

Referirea unui membru al unei variabile de tip structură se face folosind operatorul de selecție **(.)** Într-o expresie care precizează identificatorul variabilei și al câmpului.

Alocarea câmpurilor poate ridica probleme de portabilitate, deoarece organizarea memoriei depinde de sistemul de calcul.

Uniunile sunt entități care pot conține (la momente de timp diferite) obiecte de tipuri diferite. Practic, mai multe variabile sunt suprapuse în același spațiu de

memorie. Sintaxa declarației este similară cu cea a structurii, dar identificatorii declarați ca membrii reprezintă numele cu care sunt referite diferitele tipuri de variabile ce ocupă aceeași zona de memorie:

<declarație uniune> ::=

```
union <id_tip> {
    <tip_var_1> <id_var_1>;
    <tip_var_2> <id_var_2>;
    ...
    <tip_var_i> <id_var_i>;
    ...
    <tip_var_n> <id_var_n>;
} <lista identificatori de tip union>;
```

Spațiul alocat în memorie corespunde tipului cu dimensiune maxima.

Tipurile uniune sunt utile pentru conversii de date, în implementarea programelor de comunicație etc.

Tipul enumerare constă dintr-un ansamblu de constante întregi (cel puțin un element), fiecare fiind asociată câte unui identificator. Constanta unui element al enumerării este fie asociată implicit, fie explicit. Implicit, primul element are asociată valoarea 0, iar pentru restul este valoarea _precedenta+1.

Cel mai simplu program C este constituit din directive preprocesor, declarații globale și funcții. Printre funcții trebuie să existe una cu numele "main" cu care va începe executarea programului. Chiar dacă programul este organizat pe mai multe fișiere sursă, numai într-un singur fișier, numai o singura funcție poate purta numele "main". Celelalte funcții sunt subprograme definite de programator sau funcții din biblioteca de subprograme. Limbajul C nu conține funcții predefinite cum sunt cele din unitatea System a mediului Borland Pascal.

Funcțiile din bibliotecile C sunt declarate împreună cu constantele, tipurile și variabilele globale asociate, în fișiere antet, cu extensia ".h", situate în subarborele **include** al arborelui asociat mediului de programare. Operațiile de intrare-iesire necesită specificarea fișierului stdio.h, încadrat de delimitatorii < și >, într-o directivă { # include}. Fișierele antet ale programatorului vor fi încadrate folosind delimitatorul ".

O funcție C are structura:

```
<tip_rezultat> <id_functie> (<lista_parametri_formali>){
    declaratii_locale
    sevenita_instructiuni
}
```

unde <tip_rezultat> indică tipul rezultatului returnat de funcție, <id_functie> reprezintă numele (identificatorul) funcției, iar <lista_parametri_formali> constă în enumerarea declarațiilor parametrilor funcției sub forma:

```
<tip_parametru> <id_parametru> [ ,<tip_parametru> <id_parametru>]
```

Acoladele {} sunt delimitatori ce încadrează o instrucțiune compusă (bloc) alcătuită din declarații și instrucțiuni.

Secvența de instrucțiuni a funcțiilor pentru care <tip _rezultat> este diferit de tipul **void**, trebuie să conțină o instrucțiune **return**, cu sintaxa generală:
return <expresie>

Rezultatul funcției este valoarea expresiei.

Funcția **main** poate avea parametri și poate întoarce un rezultat.

III.4.2. *Funcțiile de intrare-iesire pentru consolă*

Consola sau dispozitivul standard de intrare-iesire reprezentate de tastatură (zona de date - **stdin**) și ecran (zonele de date - **stdout** și **stderr**) permit utilizatorului interacțiunea cu programul aflat în execuție. Sunt posibile operații de citire/scriere fără formatare și operații de citire/scriere cu formatare.

Operații de citire/scriere fără formatare. Acestea permit lucrul cu caractere (**char**) sau cu siruri de caractere (* **char**).

Pentru citirea unui caracter din **stdin** pot fi utilizate funcțiile: **int getchar(void)**, **int getche(void)** și **int getch(void)**, ultimele două variante nefiind prevăzute de standardul ANSI, dar sunt prezente în versiunile Borland (fișierul antet **conio.h**). Funcția **getchar** întoarce primul caracter din **stdin**, care corespunde primei taste apăsate, dar numai după apăsarea tastei **Enter**. Caracterul este transformat în întreg fără semn. În cazul unei erori sau la întâlnirea combinației **EOF** (sfârșit de fișier) funcția întoarce valoarea -1 (codificată prin **EOF**).

Funcția **getche** așteaptă apăsarea unei taste și întoarce caracterul corespunzător pe care îl afișează pe ecran (nu e nevoie de **Enter**). Funcția **getch** este similară cu **getche()**, dar nu afișează ecoul pe ecran.

Pentru scrierea unui caracter la **stdout** se utilizează funcția **int putchar (int c)** care afișează pe ecran caracterul cu codul ASCII **c**. Dacă operația reușește, întoarce caracterul afișat, iar în caz de eșec valoarea **EOF** (-1).

Pentru citirea (resp. scrierea) sirurilor de caractere se lucrează cu funcția **gets** (respectiv **puts**). Funcția cu prototipul **char *gets (char *s)** citește caractere din **stdin** și le depune în zona de date de la adresa **s**, până la apăsarea tastei **Enter**. În sir, tastei **Enter** îi va corespunde caracterul '\0'. Dacă operația reușește, funcția întoarce adresa sirului, altfel valoarea **NULL** (= 0). Funcția cu prototipul **int puts(const char *s)** afișază pe ecran sirul de la adresa **s** sau o constantă sir de caractere (secvența de caractere între ghilimele) și apoi trece la linie nouă. La succes, funcția întoarce ultimul caracter, altfel valoarea **EOF**.

Operații de citire/scriere cu formatare. La citire, formatarea specifică conversia datelor de la reprezentarea externă în reprezentarea binară. Pentru operația de scriere se efectuează conversia inversă.

Pentru citirea datelor se utilizează funcția **scanf** cu prototipul:

int scanf(const char * format [, lista_adrese_ variabile]);

iar pentru scrierea datelor se utilizează funcția **printf** cu prototipul:

int printf(const char *format, lista_valori);

Şirul de caractere format poate conține în general:

1. specificatori de format: şiruri precedate de caracterul '%' care descriu fiecare câmp așteptat;
2. caractere de spațiere: spațiu (' '), tab ('\t'), linie nouă ('\n');
3. orice alt caracter Unicode.

Fiecare variabilă din lista îi corespunde o specificație de format (tipul I.).

Funcția `scanf` întoarce numărul de câmpuri citite și depuse la adresele din listă. Dacă nu s-a stocat nici o valoare, funcția întoarce valoarea 0. Funcția `printf` întoarce numărul de octeți transferați sau EOF în caz de eșec.

Funcția `scanf` citește succesiv caractere din `stdin` pe care le interpretează prin compararea succesivă a caracterului citit cu informația curentă din sirul format. Prezența unui caracter de tip II determină citirea fără memorare a secvenței până la întâlnirea unui caracter de tip I sau III. Prezența unui caracter de tip III determină citirea fără stocare a caracterului curent de la tastatură, dacă este identic.

La scriere, caracterele de tip II și III se afișează pe ecran aşa cum apar în sirul format. Forma generală a unui descriptor pentru scriere este:

`% [flags] [width] [.prec] [lmod] type`

specificațiile dintre [și] fiind opționale. Elementele de mai sus au următoarea semnificație:

flags - poate fi unul dintre semnele: +, -, 0, spațiu, #. Semnele au următoarea semnificație:

- : aliniere la stânga a argumentului în cadrul câmpului;

+ : numerele vor fi obligatoriu tipărite cu semn;

0 : indică completarea la stânga cu zerouri (la numere);

spațiu: dacă primul caracter nu este semnul , se va afișa un spațiu;

width: este un număr care specifică lățimea minima a câmpului.

Argumentul corespunzător va fi afișat pe un câmp cu latine cel puțin `width`. Dacă sunt mai puține caractere de scris, se va completa câmpul cu spatii la stânga (implicit) sau la dreapta, dacă s-a specificat flag-ul -. Dacă s-a specificat flagul 0, se va completa la stânga cu zero. Dacă `width` este caracterul *, atunci lățimea este dată de următorul argument din listă (trebuie să fie neapărat un int).

prec: este un număr care specifică precizia de scriere; pentru %s prec indică numărul maxim de caractere ce se va scrie; pentru %e, %E și %f prec indică numărul de zecimale; pentru %g și %G prec indică numărul de cifre semnificative, iar la descriptorii pentru întregi indică numărul minim de cifre. Dacă prec este *, atunci se consideră că lățimea de scriere este dată de următorul argument din listă, care trebuie să fie de tip int.

lmod: este un specificator de lungime care corespunde unui argument short sau `unsigned short` (h), long sau `unsigned long` (l), respectiv `long double` (L).

type: este descriptorul propriu-zis. Se utilizează următoarele caractere: d, i (int) - notație zecimală cu semn; 0 (int) - notație în baza 16 fără semn; x, X (int) - notație în baza 16 fără semn cu abcdef pentru x și ABCDEF pentru X; u (int) - notație zecimală fără semn; c (int) - un caracter; s (char *) - sir de caractere terminat cu '\0'; f (double) - numărul în virgulă mobilă cu format standard; e, E

(double) - numărul în virgulă mobilă cu format exponential; g, G (double) - în loc de f, e, E; p (void *) - se tipărește argumentul ca adresă; % - se tipărește %.

Forma generală a unui descriptor pentru citire este:

% [*] [width] [lmod] type

unde:

* - suprimă atribuirea următorului câmp din stdin la următoarea variabilă; width, lmod - ca mai sus;

type - descrie tipul de conversie. Cele mai importante specificații de conversie sunt: d (int *) - întreg zecimal; i (int *) - întreg oarecare (zecimal, octal sau hexa); o (int *) - întreg octal; u (unsigned int *) - întreg zecimal fără semn; x (int *) - întreg hexa, c (char *) - caractere; s (char *) - sir de caractere (se va încheia cu '\0'); e, f, g (float *) - numere în virgulă mobilă; p (void *) - valoarea unei adrese aşa cum e tipărită de printf.

În descrierea de mai sus, între paranteze se indică tipul argumentului supus operației de intrare-iesire. Notația *tip* * înseamnă adresa unei locații de **tipul tip**.

III.4.3. Operatori și expresii

Operatorii sunt simboluri care descriu operații efectuate asupra unor variabile sau constante (numite generic operanți). O combinație corectă de operatori, variabile, constante, apeluri de funcții reprezintă o expresie. Pentru construcția expresiilor, limbajul C oferă o gamă foarte largă de operatori.

Operatorul de atribuire. Operatorul de atribuire (=) permite crearea unei expresii de forma:

<variabila> = <expresie>

ce se evaluatează de la dreapta la stânga. După evaluarea membrului drept, valoarea rezultată este înscrisă în <variabila>, iar întreaga construcție are valoarea variabilei după înscriere.

Operatori aritmetici. Operatorii aritmetici sunt: + (adunare), - (scădere), * (înmulțire), / (împărțire), % (împărțire modulo împărțitor). Ordinea operațiilor este cea binecunoscută, dar se pot utiliza paranteze pentru schimbarea ordinii operațiilor. Pentru scrierea instrucțiunii de atribuire <variabila> = <variabila> + 1; se poate utiliza forma prescurtată <variabila>++, operatorul ++ fiind numit *operator de incrementare*. Există, de asemenea, și un operator de decrementare (--): <variabila>-- ce este echivalentul instrucțiunii: <variabila> = <variabila> - 1;

Operatori logici și relationali. Pentru scrierea expresiilor booleene se utilizează operatorii logici și operatorii relaționali. Există trei operatori logici: || (SAU logic - SAU INCLUSIV), && (SI logic), ! (NU logic). Operatorii relaționali întâlniți în limbajul C sunt următorii: < (mai mic strict), > (mai mare strict), <= (mai mic sau egal), >= (mai mare sau egal), == (egal cu), != (diferit de). Ori de câte ori relația este falsă se generează valoarea 0, valoarea 1 fiind generată atunci când relația este adevărată. Trebuie evidențiat că operatorii aritmetici au prioritate față de operatorii relaționali.

Operatori la nivel de bit. Operatorii la nivel de bit se pot aplica operanzilor de tip întreg (char, int, short, long, cu sau fără semn): & (SI logic la nivel de bit),

| (SAU logic la nivel de bit), ^ (SAU exclusiv la nivel de bit), << (deplasare stânga), >> (deplasare dreapta) și ~ (negare la nivel de bit).

Operatori de atribuire combinați. Pentru realizarea atribuirii

<variabila> = <variabila> <operator> <var_sau_const>;

se pot utiliza operatorii de atribuire combinați: += (atribuire cu adunare), -= (atribuire cu scădere), *= (atribuire cu înmulțire), /= (atribuire cu împărțire), %=(atribuire cu împărțire modulo); expresia fiind scrisă prescurtat:

<variabila> <operator>= <var_sau_const>;

Operatorul virgulă. În limbajul C, virgula (,) este un operator binar, care leagă expresii oarecare. Construcția <expresie_1>, <expresie_2> este una corectă, constând din evaluarea celor două expresii, în ordinea în care apar, valoarea întregii construcții fiind dată de valoarea lui <expresie_2>. Asocierea operatorului virgulă se face de la stânga la dreapta, astfel încât o expresie de forma e1,e2,e3 este echivalentă cu: (e1, e2), e3.

Operatorul condițional (?:) Operatorul condițional este o construcție decizională a limbajului C care are următoarea formă generală: <Expresie-booleana> ? <expresie_1> : <expresie_2>; având următorul înțeles: Dacă <Expresie-booleana> este adevărată, atunci întreaga expresie condițională are valoarea <expresie_1>, în caz contrar, valoarea expresiei condiționale fiind valoarea <expresie_2>.

Alți operatori: În această categorie includem operatorii specifici tipului referință, operatorii pentru selectarea elementelor unui tip de date structurat, precum și operatorii introdusi de extensia C++.

III.4.4. Instrucțiuni C

Cea mai simplă instrucționă C este instrucționă <expresie>; ce oferă un echivalent în C pentru următoarele instrucționări Pascal: atribuire, apel de procedură, instrucționă vidă. O secvență de instrucționări încadrată de acolade este considerată ca o singură instrucționă și este numită instrucționă compusă sau bloc. Spre deosebire de instrucționă compusă a limbajului Pascal, instrucționă compusă din limbajul C poate conține atât declarații, cât și instrucționări, declarațiile fiind poziționate la începutul blocului. În mod implicit, identificatorii declarați în blocul delimitat de acolade, au ca domeniu de vizibilitate blocul, iar timpul de viață este limitat la timpul de executare a blocului.

Programarea unei structuri decizionale, în C, poate fi realizată folosind: instrucționă if...else ; operatorul condițional (?:) și instrucționă switch. Sintaxa instrucționii if...else este:

<instrucționă_if> ::=

if (<Expresie>) <Instrucționă_T>; |

if (<Expresie>) <Instrucționă_T> else <Instrucționă_F>

unde: <Expresie> are o valoare de tip scalar reprezentând o construcție de tip expresie, <Instrucționă_T> reprezintă o instrucționă C care se va executa când <Expresie> are o valoare nenulă (adevărat), iar <Instrucționă_F> reprezintă acea instrucționă C ce se va executa pentru valoare 0 (false) a lui <Expresie>.

Conform celor de mai sus, construcția: e1 ? e2 : e3; poate înlocui instrucțiunea if...else: if (e1) e2; else e3;

Atunci când o selecție multiplă este controlată de valoarea unei singure expresii, se poate utiliza instrucțiunea **switch**, un pseudo-echivalent C a construcției Pascal **case**.

Sintaxa instrucțiunii **switch** este:

```
<instructiune_switch> ::= switch ( <expresie> ) {  
    case <const_1> : <lista_instructiuni> [ break; ]  
    case <const_2> : <lista_instructiuni> [ break; ]  
    ...  
    [ default:] <lista_instructiuni> [ break ; ]  
}
```

unde: <expresie> este o expresie cu valoare întreagă (tip întreg sau enumerare); <const_1>, <const_2>, ... sunt constante de selecție, cu valori distincte, convertibile la tipul expresiei <expresie>, iar <lista_instructiuni> este o secvență de instrucțiuni C.

Fiecare etichetă **case** indică o singură constantă, dar se pot asocia mai multe etichete **case**, scrise consecutiv, pentru aceeași secvență de instrucțiuni.

Instrucțiunea **break** întrerupe lista de instrucțiuni și duce la încheierea instrucțiunii **switch**. Dacă valoarea expresie nu apare în lista constantelor de selecție, se execută instrucțiunile asociate etichetei **default**, dacă există.

Programarea ciclurilor poate fi realizată folosind instrucțiunile de ciclare: ciclul cu test inițial (instrucțiunea **while**), ciclul cu test final (instrucțiunea **do...while**) și ciclul cu test inițial și contor (instrucțiunea **for**).

Forma instrucțiunii **while** este:

```
while (<expresie>) <instructiune>.
```

În particular, <instructiune> poate fi chiar instrucțiunea vidă.

Sintaxa instrucțiunii **do..while** este:

```
do <instructiune> while (<expresie>);
```

Instrucțiunea dintre **do** și **while** se execută cel puțin o dată și se repetă cât timp <expresie> este compatibilă cu valoarea logică “adevărat”.

Instrucțiunea **for**, oferă cea mai compactă metodă pentru scrierea ciclurilor cu test inițial și are o definiție care îi extinde domeniul de aplicare față de alte limbaje de programare. Forma instrucțiunii **for** este:

```
for ( <expresie_1> ; <expresie_3> ; <expresie_3> ) <instructiune>
```

și are efectul similar cu al secvenței:

```
<expresie_1>; while (<expresie_2>) { <instructiune> <expresie_3>; }
```

Cele trei expresii dintre paranteze pot fi toate vide, caz în care avem de-a face cu un ciclu infinit.

Întreruperea necondiționată a unei secvențe de instrucțiuni și continuarea dintr-un alt punct al programului este posibilă prin utilizarea instrucțiunilor de salt: **goto** (salt la o instrucțiune etichetată), **break** (în contextul instrucțiunii switch cât și în instrucțiunile de ciclare pentru a determina ieșirea forțată din ciclu, indiferent de valoarea condiției de ciclare) și **continue** (în cadrul blocului instrucțiunilor de ciclare pentru a întrerupe execuția iterației curente). În cazul instrucțiunilor **while** și **do..while**, instrucțiunea **continue** determină activarea testului condiției de ciclare, iar pentru instrucțiunea **for** se va continua cu evaluarea, În această ordine, expresiilor <expresie_3>, <expresie_2>.

BIBLIOGRAFIE

1. Albeanu G., *Algoritmi și limbaje de programare*, Editura Fundației România de Mâine, București, 2000.
2. Albeanu G., Luminita Radu, *Algoritmica și programare în Pascal*, Editura Fundației România de Mâine, București, 2001.
3. Knuth D., *Arta programării calculatoarelor*, vol. I, *Algoritmi fundamentali*, Editura Teora, 2000.
4. Livovschi L., Georgescu H., *Analiza și sinteza algoritmilor*, Editura Științifică și Enciclopedică, 1974.
5. Albeanu G., *Tehnici de programare, Lucrări practice de programarea calculatoarelor*, Editura Fundația România de Mâine, 2003.
6. Bârză S., *Culegere de probleme de algoritmică și programare*, vol I, *Programare statică*, Editura Universității București, 2001.
7. Bârză S., *Algoritmica și programare. Note de curs*, vol. I, Programare statică, Editura Universității București, 2001.

ARHITECTURA SISTEMELOR DE CALCUL

Prof. univ. dr. GRIGORE ALBEANU

I. BAZELE NUMERICE ȘI ARHITECTURALE ALE SISTEMELOR DE CALCUL

I.1. Noțiuni fundamentale privind prelucrarea informațiilor

I.1.1. *Informație. Sisteme de numerație. Coduri*

Pentru prelucrarea informației, omul lucrează în *sistemul de numerație zecimal*, folosind cele 10 simboluri: 0, 1, 2, ..., 9, numite *cifre* (eng. *digits*). Pentru modelarea informației de tip text omul a inventat scrisul și deci semne speciale pentru exprimarea textelor.

Pentru domeniul calculatoarelor, cele mai importante sisteme de numerație sunt: binar, octal și hexazecimal. *Sistemul binar* folosește baza de numerație 2 ($0 + 1 = 1 + 0 = 1; 0 + 0 = 0; 1 + 1 = (10)_2 = 2$ (în baza 10)). *Sistemul octal*, cu baza opt, folosește simbolurile: 0, 1, ..., 7. Baza de numerație a *sistemului hexazecimal* este 16, iar simbolurile folosite sunt: 0, 1, ..., 9, a, b, c, d, e, f. Indiferent de sistemul de numerație utilizat, modul de realizarea operațiilor aritmetice este același. Procesul privind transformarea reprezentărilor exprimate în sisteme de numerație se numește *conversie*. Referitor la sistemele de numerație, descriem câteva metode de conversie și operații de calcul.

a) *Conversia unui număr real într-o bază de numerație b* ($b \geq 2$)

Codificarea unui număr real într-o bază de numerație b se bazează pe operațiile de împărțire și înmulțire [1] aplicate numerelor întregi. Pentru a converti un număr real format din parte întreagă și parte fraționară, din scrierea zecimală, în baza b ($b \geq 2$), se procedează astfel:

1. se împarte (conform teoremei împărțirii cu rest) la b , partea întreagă și cătul obținute după fiecare împărțire, până se obține câtul zero. Rezultatul conversiei este constituit din resturile obținute, luate în ordine inversă apariției acestora.
2. se înmulțește cu b , partea fraționară și toate părțile fraționare obținute din produsul anterior, până când partea fraționară este *nulă* sau a fost obținut numărul de cifre dorit. Rezultatul conversiei părții fraționare este constituit din părțile întregi ale produselor, luate în ordinea apariției.

Conversia binară a numărului zecimal 24,25 este 11000,01. Numărul zecimal 2002,2003 este reprezentat în sistem octal cu 10 poziții în partea fraționară ($b = 8$) prin sirul: 3722,1464011651. Numărul zecimal 1961,25 este reprezentat în format hexazecimal ($b = 16$) prin sirul: 7A9,4.

b) *Conversia unui număr dintr-o bază de numerație b (b ≥ 2) în zecimal*

Pentru a transforma un sir de simboluri ale sistemului de numerație în baza b , în zecimal, se va calcula suma produselor dintre cifra corespunzătoare (din sir) și baza ridicată la puterea specificată de poziția acesteia. Trebuie observat că pozițiile sunt indicate:

1. pentru partea întreagă, de la dreapta la stânga, prin numerele $0, 1, \dots$ și.a.m.d.
2. pentru partea fracționară, de la stânga la dreapta, prin numerele: $-1, -2, \dots$ și.a.m.d.

Şirul binar $01100110101,10101$ corespunde numărului zecimal: $821,65625 (= 1 \times 2^9 + 1 \times 2^8 + 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5})$.

Şirul octal $765,567$ corespunde numărului zecimal: $501,732421875$ (Verificați după modelul anterior!). Şirul hexazecimal $3A5,4$ reprezintă numărul zecimal $933,25$.

c) *Conversia din binar în octal, hexazecimal și invers*

Deoarece $8 = 2^3$, iar $16 = 2^4$, folosind proprietățile de calcul, se obține o strategie de conversie automată între aceste sisteme. Conversia *binar → octal*, respectiv *octal → binar* folosește corespondența:

Octal:	0	1	2	3	4	5	6	7
Binar:	000	001	010	011	100	101	110	111

Conversia *binar → hexazecimal*, respectiv *hexazecimal → binar*, folosește corespondența:

Hexazecimal:	0	1	2	3	4	5	6	7
Binar:	0000	0001	0010	0011	0100	0101	0110	0111
Hexazecimal:	8	9	A	B	C	D	E	F
Binar:	1000	1001	1010	1011	1100	1101	1110	1111

Şirul binar:

$1101101101110110101011101010110101010101010101010111$

se va “traduce” în sirul octal: 1555672535325252527 , respectiv în sirul hexazecimal: $6DBAAEB555557$. Se observă rolul sistemului binar ca sistem intermediu de conversie pentru sistemele de numerație în care baza este o putere a numărului doi. Aceasta este o proprietate generală și se aplică tuturor sistemelor de numerație ce au baza o putere a unui număr.

d) *Operații aritmetice în binar, octal și hexazecimal*

Operațiile aritmetice cu numere binare, octale, respectiv hexazecimale se efectuează similar operațiilor cu numere zecimale. La adunare va interveni transportul către ordinul superior, la scădere va interveni împrumutul de la ordinul superior, iar înmulțirea se va desfășura prin totalizarea unor produse partiiale, analog modului de calcul zecimal.

Operațiile sunt efectuate conform următoarelor reguli:

Tabla adunării binare:

+	0	1
0	0	1
1	1	(10)

Tabla înmulțirii binare:

x	0	1
0	0	0
1	0	1

Tabla adunării octale:

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	(10)
2	2	3	4	5	6	7	(10)	(11)
3	3	4	5	6	7	(10)	(11)	(12)
4	4	5	6	7	(10)	(11)	(12)	(13)
5	5	6	7	(10)	(11)	(12)	(13)	(14)
6	6	7	(10)	(11)	(12)	(13)	(14)	(15)
7	7	(10)	(11)	(12)	(13)	(14)	(15)	(16)

Tabla înmulțirii octale:

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	(10)	(12)	(14)	(16)
3	0	3	6	(11)	(14)	(17)	(22)	(25)
4	0	4	(10)	(14)	(20)	(24)	(30)	(34)
5	0	5	(12)	(17)	(24)	(31)	(36)	(43)
6	0	6	(14)	(22)	(30)	(36)	(44)	(52)
7	0	7	(16)	(25)	(34)	(43)	(52)	(61)

Pentru sistemul de numerație hexazecimal se pot construi table similare (Exercițiu).

Fie A și B două mulțimi nevide și B^+ mulțimea sirurilor (numite și *cuvinte* peste B) formate cu elemente din B (în calitate de *alfabet*). Practic, un cuvânt peste alfabetul B este o secvență finită de simboluri din B: $w = a_1a_2...a_k$, $k \geq 1$, $a_i \in B$, $1 \leq i \leq k$. Numărul k reprezintă *lungimea cuvântului* w . Mulțimea tuturor cuvintelor peste B, la care se adaugă cuvântul de lungime zero (cuvântul vid), identificat prin λ , se notează cu B^* , adică $B^* = B^+ \cup \{\lambda\}$.

Prin *codificare* a elementelor mulțimii A cu ajutorul elementelor mulțimii B înțelegem determinarea unei funcții injective, numită *cod*, definită pe mulțimea

A cu valori în mulțimea B^+ . Codurile în care sunt reprezentate numai numere se numesc *coduri numerice*, iar cele care cuprind numere, literele și celealte semne se numesc *coduri alfanumerice*.

Dacă mulțimea A are n elemente, ne interesează determinarea *lungimii codului*, adică lungimea maximă a secvenței din B^+ pentru a se putea codifica toate cele n obiecte. Dacă toate cuvintele din B^+ au aceeași lungime codul se numește *uniform*. Pentru cazul sistemelor de calcul, mulțimea B este formată din simbolurile unui sistem de numerație, iar lungimea codului depinde de natura obiectelor mulțimii A [1].

Atfel spus, *un cod este un set de simboluri elementare împreună cu o serie de reguli potrivit cărora se formează aceste simboluri. Codificarea reprezintă procesul de stabilire a unui cod.* Codurile pot fi alfanumerice și grafice (folosind imagini). Un exemplu de cod nenumeric, foarte popular, este codul de bare (eng. *bar code*), utilizat în supermagazine.

I.1.2. Reprezentarea datelor în sistemele de calcul

Primele mașini de calcul (în special mecanice) au folosit sistemul de numerație zecimal. La baza tehnicii calculului electronic au stat rezultatele obținute în cadrul teoriei informației și în domeniul circuitelor electronice active. Cercetările întreprinse, au arătat că sistemul de numerație potrivit calculatoarelor digitale este cel binar, adică orice informație, oricât de complexă este, poate fi exprimată prin informații elementare [1].

Informația elementară este numită *bit* (engleză *binary digit*). Un bit este descris prin una din cifrele binare: 0, 1. Biți se pot grupa câte 8, 16, 32 etc. formând un *octet* (engleză *byte*), *cuvânt* (engleză *word*), *cuvânt dublu* (engleză *double word*) etc. Informația modelată și prelucrată de calculator este prezentată sub forma unui sir de cifre binare. Aceste siruri fără a avea vreo semnificație se numesc *date*. Într-un calculator, informația reprezentată codificat formează *mulțimea datelor*.

Pentru a modela sistemul de numerație binar, trebuie utilizat un mecanism cu două stări (un comutator). Prin urmare, calculatorul poate fi privit ca un ansamblu de circuite electrice, electronice și părți mecanice, ansamblu numit *hardware*¹. Prelucrarea informației este realizată de *programe*, compuse din comenzi numite *instrucțiuni*. Totalitatea programelor poartă numele de *software*². Se pot efectua operații de evaluare sau de transfer (cazul rețelelor de calculatoare).

¹ Cuvântul *hardware* descrie totalitatea resurselor fizice ale unui sistem de calcul. Conform dicționarului de informatică [DINF1981], *hardware* (din limba engleză) reprezintă un "termen general desemnând circuitele, dispozitivele și echipamentele componente ale unui sistem de calcul", adică "toate unitățile fizice existente, cu funcții bine determinate, în cadrul unui sistem de calcul; funcțiile sale, specificate de către fabricant, sunt la dispoziția utilizatorului care le poate exploata cum dorește."

² Cuvântul *software* descrie atât programele de bază cât și pe cele aplicative. Conform [DINF1981], *software* (din limba engleză) reprezintă un "termen utilizat pentru a desemna: a) totalitatea programelor cu care este echipat un sistem de calcul; b) preocupările

Sistemele de calcul utilizează cel mai frecvent coduri alfanumerice cu 7, 8 și 16 cifre binare care permit reprezentarea a 128 (*ASCII*), 256 (*ASCII extins*), respectiv 65536 (*Unicode*) obiecte (cifre, litere, caractere speciale). Codul *ASCII* (engleză *American Standard Code for Information Interchange*), pronunțat “as-key” este cel mai popular cod. *Unicode* este standardul recent introdus pentru acoperirea lingvistică la nivel mondial, în elaborarea aplicațiilor în tehnologia Java.

Codurile numerice oferă posibilitatea reprezentării numerelor folosind sistemul binar. Reprezentarea numerelor în acest sistem se face în mai multe forme, în funcție de mulțimea căreia îi aparțin numerele, operațiile aritmetice fiind efectuate de către dispozitive aritmetice specializate. Reprezentarea numerelor naturale se realizează pe un număr fix de poziții binare (de regulă 8, 16, 32, sau 64) prin conversia numărului zecimal în baza 2. Această reprezentare este numită *reprezentare aritmetică*. Prin utilizarea a n poziții binare ($n \geq 1$) se pot reprezenta aritmetic toate numerele naturale din plaja $0 \dots 2^n - 1$ (Verificați acest lucru prin metoda de conversie *binar → zecimal*).

Reprezentarea numerelor întregi - numită și *reprezentare algebrică* - este asemănătoare reprezentării numerelor naturale, cu deosebirea că prima poziție este ocupată de semnul numărului întreg. Reprezentarea numerelor întregi negative se poate realiza în trei forme: *codul direct* (care coincide cu reprezentarea algebrică a numerelor întregi, cu prima poziție destinată semnului S: S = 0 dacă numărul este pozitiv, respectiv S = 1, dacă numărul este negativ); *codul invers* (se obține prin schimbarea fiecărei cifre - din 0 în 1, respectiv din 1 în 0 - a codului direct); *codul complementar* (se obține prin adunarea numărului 1 la cifra cea mai puțin semnificativă a reprezentării în cod invers).

Cunoscând numărul n ($n \geq 1$) de poziții binare pe care se reprezintă un număr întreg în forma algebrică, plaja numerelor care admit o reprezentare în cod complementar este $-2^{n-1} \dots 2^{n-1} - 1$. Codul complementar este potrivit pentru aplicații deoarece operația de scădere este transformată în operație de adunare conform formulei $a - b = a + (-b)$, iar scăzătorul este reprezentat în cod complementar. Dacă rezultatul scăderii este un număr negativ, acesta este reprezentat tot în cod complementar. Dacă apare transport de la poziția alocată semnului, acesta se ignoră. Acest aspect va face ca, în cadrul reprezentării în cod complementar pe 16 biți, să se obțină:

$$\begin{aligned}32767+1 &= -32768; \\32767 + 2 &= -32767; \\20000+25000 &= -20536.\end{aligned}$$

Numerele reale (mai precis numerele raționale) se reprezintă sub formă fracționară prin intermediul codificării în virgulă mobilă (standardul *IEEE 754*). Se disting: reprezentarea în virgulă mobilă cu precizie simplă, pe 32 biți: *float* (C, C++, Java) sau *single* (Pascal) și, reprezentarea în precizie dublă, pe (64 biți): *double* (C, C++, Java, Pascal).

corespunzătoare realizării produselor program și, în cel mai larg sens, analizei și cercetărilor efectuate în raport cu activitățile conexe realizării programelor."

Operațiile aritmetice cu numere în virgulă mobilă sunt, fie realizate software, fie prin intermediul unor dispozitive electronice specializate (procesoare de calcul în virgulă mobilă). Actualele procesoare numerice includ și reprezentări în precizie extinsă : *long double* (C, C++) sau *extended* (Pascal).

Fie q și b numere naturale nenule, iar $x \in \mathbf{R}$ dat. Numim *reprezentare cu virgulă mobilă a numărului* x , *cu excesul* q , *în baza* b , perechea (e,f) cu semnificația:

$$x = f \cdot b^{e-q} \text{ și } |f| < 1.$$

Componenta e se numește *parte exponentială*, iar componenta f , *parte fracționară* [1]. Dacă reprezentarea impune utilizarea a $p \geq 1$ ($p \in \mathbb{N}^*$) cifre în baza b atunci

$$-b^p \leq b^p f < b^p.$$

Notăm cu $e(x)$ (respectiv $f(x)$) cantitatea e (respectiv f) pentru a specifica numărul x luat în considerare.

Fie x, y numere reale, q și b ca mai sus, dar fixate. Relația de ordine " $>$ " este definită astfel: $x > y \Leftrightarrow e(x) > e(y)$ sau ($e(x) = e(y)$ și $f(x) > f(y)$).

Despre un număr cu virgulă mobilă (e, f) se spune că este reprezentat normalizat dacă cifra cea mai semnificativă a reprezentării componentei f este nenulă, adică

$$1/b \leq |f| < 1 \text{ pentru } f \neq 0 \text{ sau}$$

componenta e are cea mai mică valoare posibilă pentru $f = 0$.

Fie e_{\min}, e_{\max} numere întregi fixate. *Mulțimea numerelor reale cu virgulă mobilă reprezentabile exact, în baza b , cu excesul q , folosind exact p poziții pentru partea fracționară este* $F_{p,q} = \{x \in \mathbb{Q} / x = f \cdot b^{e-q}, e_{\min} \leq e-q \leq e_{\max}, f = \pm(c_1 b^{-1} + c_2 b^{-2} + \dots + c_p b^{-p}), c_i \in \{0, 1, \dots, b-1\}, i = 1, 2, \dots, p\}$. Fie x un număr real. Spunem că x este situat în domeniul de valori numai dacă $x \in F_{p,q}$.

Exponentul minim e_{\min} este un număr întreg negativ, iar exponentul maxim e_{\max} este un număr întreg pozitiv. De obicei se memorează $b^p f$ folosind una dintre reprezentările cu semn pentru numere întregi (cod invers sau cod complementar).

Observații:

1. Numerele $k = b^{e_{\min}-1}$ și $K = b^{e_{\max}}(1 - b^{-p})$ sunt: *cel mai mic număr pozitiv reprezentabil*, respectiv *cel mai mare număr reprezentabil*. $-K$ este *cel mai mic număr reprezentabil*. Dacă în timpul operațiilor aritmetice rezultă numere în valoare absolută mai mici decât k (resp. mai mari decât K) se spune că s-au produs *depășiri aritmetice* inferioare (respectiv, superioare).
2. Nu orice număr rațional x ($x \in \mathbb{Q}$) poate fi reprezentat, chiar dacă $k \leq |x| \leq K$. De exemplu, $x = 0,1$ număr reprezentabil exact în baza 10, nu se poate reprezenta exact în baza 2. Aceasta face ca expresia: $3/5 * 5 - 3$ să fie evaluată de către mediul de programare Turbo Pascal, la valoarea 2.1684043450E-19, adică

$$0,000000000000000000000021684043450,$$

un număr foarte mic, dar nenul.

3. Dacă $x \in F_{p,q}$ atunci $-x \in F_{p,q}$. Totuși, pentru $x \in F_{p,q}$ se poate întâmpla ca $1/x$ să nu fie reprezentabil sau să nu fie în domeniul de valori.

4. Sistemele de calcul actuale utilizează reprezentarea normalizată. În acest caz pentru $f \neq 0$ avem $c_1 \neq 0$. Numărul 0 are o reprezentare specială care depinde de tipul sistemului de calcul.

Fie $x \in R$, valoarea reprezentării lui x aparținând mulțimii $F_{p,q}$, se notează prin $fl(x)$ și se definește astfel: *numărul reprezentabil exact, cel mai apropiat de x (metoda rotunjirii) sau cel mai apropiat număr reprezentabil, cu modul mai mic decât $|x|$ (metoda trunchierii).*

Rezultă:

$$\text{Oricare } x \in R, x \neq 0, \frac{|x - fl(x)|}{|x|} \leq cb^{1-p}, \text{ unde } c = 0.5 \text{ în cazul metodei}$$

rotunjirii și $c = 1$ în cazul metodei trunchierii.

Membrul stâng al inegalității de mai sus se numește *eroarea relativă* a reprezentării numărului x . Acest rezultat este foarte important pentru interpretarea rezultatelor obținute prin prelucrarea numerică a datelor.

Observații:

1. Sistemele de calcul bazate pe reprezentarea cu rotunjire sunt de preferat celor bazate pe trunchiere, deoarece în cazul trunchierii, eroarea relativă poate fi de până la o cifră a bazei în ultima poziție.
2. Cu cât lungimea reprezentării părții fracționare (p) este mai mare, cu atât eroarea relativă a reprezentării este mai mică. Adică precizia reprezentării este mai mare. De aceea, de multe ori, numărul p se mai numește (prin abuz de limbaj) și *precizia reprezentării*.

Fie p, b, e și q fixate. Distanța absolută dintre două numere consecutive din $F_{p,q}$ este constantă și egală cu b^{e-q-p} . Această distanță crește însă cu $e-q$ și b pentru p fixat. Distanța relativă definită ca $b^{p-p}/|f|$ scade o dată cu creșterea părții fracționare. Astfel se poate afirma, din nou, că sistemele de calcul preferate sunt cele care utilizează baza 2.

Dacă $f = 1/b$ atunci se obține distanța relativă maximă, $\varepsilon_M = b^{1-p}$. În cazul multor sisteme de calcul $\varepsilon_M = 2\varepsilon_u$ unde ε_u este cel mai mic număr pozitiv reprezentabil pentru care $fl(1+\varepsilon_u) > 1$.

Pentru sistemele de calcul pentru care $b = 2$, în anul 1985, a fost elaborat standardul IEEE 754. Un număr în virgulă mobilă este reprezentat cu ajutorul a $c+m+1$ biți (un bit pentru semn, m biți pentru partea fracționară și c biți pentru reprezentarea lui $e-q$ (numit și caracteristică)) astfel încât numărul $c+m+1$ să fie un multiplu al lungimii cuvântului sistemului de calcul.

Exemplificăm implementarea reprezentărilor în virgulă mobilă IEEE 754: precizie simplă (*Single*), precizie dublă (*Double*) și precizie extinsă (*Extended*) pentru calculatoare compatibile IBM-PC, aşa cum sunt utilizate de limbajul *Pascal*. *NaN*, respectiv *Inf* sunt coduri care descriu nedeterminarea, respectiv ∞ (infinit).

Reprezentarea *Single* utilizează 32 biți repartizați astfel: b_{31} - bit de semn (notat în continuare cu s), biții $b_{23}-b_{30}$ pentru memorarea caracteristicii (c), iar biții b_0-b_{22} pentru reprezentarea părții fracționare (f). Valoarea $fl(x)$ se obține conform regulilor:

- S1: Dacă $0 < c < 255$ atunci $fl(x) = (-1)^s \cdot 2^{(c-127)} \cdot (1.f)$.
 S2: Dacă $c = 0$ și $f \neq 0$ atunci $fl(x) = (-1)^s \cdot 2^{(c-126)} \cdot (0.f)$.
 S3: Dacă $c = 0$ și $f = 0$ atunci $fl(x) = (-1)^s \cdot 0$.
 S4: Dacă $c = 255$ și $f = 0$ atunci $fl(x) = (-1)^s Inf$.
 S5: Dacă $c = 255$ și $f \neq 0$ atunci $fl(x) = NaN$.

Reprezentarea *Double* utilizează 64 biți repartizați similar: b_{63} - bit de semn, biții $b_{52}-b_{62}$ - caracteristica, iar biții b_0-b_{51} pentru reprezentarea părții fracționare. Valoarea numărului reprezentat astfel, se obține după cum urmează:

- D1: Dacă $0 < c < 2047$ atunci $fl(x) = (-1)^s \cdot 2^{(c-1023)} \cdot (1.f)$.
 D2: Dacă $c = 0$ și $f \neq 0$ atunci $fl(x) = (-1)^s \cdot 2^{(c-1022)} \cdot (0.f)$.
 D3: Dacă $c = 0$ și $f = 0$ atunci $fl(x) = (-1)^s \cdot 0$.
 D4: Dacă $c = 2047$ și $f = 0$ atunci $fl(x) = (-1)^s Inf$.
 D5: Dacă $c = 2047$ și $f \neq 0$ atunci $fl(x) = NaN$.

Reprezentarea *Extended* utilizează 80 biți repartizați astfel: b_{79} - bit de semn, biții $b_{64} - b_{78}$ - caracteristica, bitul b_{63} este 0 sau 1 (notat mai jos cu i), iar biții b_0-b_{62} pentru reprezentarea părții fracționare. Valoarea numărului în această reprezentare se obține după cum urmează:

- E1: Dacă $0 \leq c < 32767$ atunci $fl(x) = (-1)^s \cdot 2^{(c-16383)} \cdot (i.f)$.
 E2: Dacă $c = 32767$ și $f = 0$ atunci $fl(x) = (-1)^s Inf$.
 E3: Dacă $c = 32767$ și $f \neq 0$ atunci $fl(x) = NaN$.

În cazul transferului datelor între sisteme de calcul pot apărea erori. De aceea se utilizează coduri de control cu posibilitatea detectării și corectării erorilor. Mai precis, se atașează cifre binare (de control) la emisia mesajului, recepția fiind responsabilă de controlul modului de respectare a corectitudinii mesajului. Cele mai utilizate procedee pentru detectarea erorilor sunt: *codurile pentru controlul parității și codurile polinomiale ciclice*.

Din cele de mai sus rezultă că datele sunt reprezentarea fizică (prin intermediul codului) a entităților din care este compusă informația (cifre, litere, semne speciale, imagini, sunete etc.) pentru ca aceasta să poată fi stocată, prelucrată sau transmisă.

Din punct de vedere logic, unei date i se asociază un identificator (simbol sau nume pentru diferențierea de alte date și pentru referirea acestaia). În vederea prelucrării, datelor le sunt asociate atribute precum: tipul datei (numeric: întreg sau real; logic; sir de caractere; enumerare, adresă etc.), precizia reprezentării interne și modul de vizualizare (poziție, aliniere, corpul simbolului, dimensiunea etc). În timpul prelucrării unele date își păstrează valoarea inițială (se numesc *constante*), iar altele se modifică în timp (se numesc *variabile*).

I.2. Generații de calculatoare. Arhitectura sistemelor de calcul

Tehnica de calcul a evoluat de la *abac* până la *supercalculatoarele actuale* (sisteme hipercub și.a.). În această evoluție se disting următoarele etape: etapa dispozitivelor mecanice și electromecanice și etapa calculatoarelor electronice. Primul calculator electronic comercial a fost sistemul ENIAC. Aproximativ în acea perioadă, John von Neumann (1944) a stabilit că un sistem de calcul trebuie să

asigure următoarele funcții: *memorare, comandă și control, prelucrare, intrare-iesire*.

Etapele de dezvoltare a calculatoarelor electronice sunt cunoscute sub numele de *generații de calculatoare*. Până în prezent au fost parcurse cinci generații: *generația tuburilor electronice* (programe binare sau cablate, suport informațional: cartela perforată sau banda de hârtie perforată), *generația tranzistorilor și diodelor* (apar suporturile magnetice, apar limbajele de programare: FORTRAN: eng. *FORmula TRANslation*, COBOL: eng. *COmmon Business-Oriented Language*, ALGOL: eng. *ALGOrithmic Language* și.a., apar primele sisteme de operare), *generația circuitelor integrate* (apare conceptul de *firmware*, apar sisteme de operare evoluționate pe disc (DOS: eng. *Disk Operating System*), apar noi limbaje de programare: PL/I, Pascal, LISP: eng. *LIS Processing*, BASIC), *generația VLSI*: eng. *Very Large Scale Integration* (apar microprocesoarele, noi tipuri de arhitecturi: microcalculatoarele (de exemplu cele compatibile IBM PC – eng. *Personal Computer*, stațiile MAC etc.), sisteme de operare precum CP/M, DOS (de la IBM sau Microsoft); se dezvoltă rețelele de calculatoare; apar limbajele de programare orientată obiect) și *generația inteligenței artificiale* (în continuă dezvoltare) cu trimiteri către viitoarele sisteme neurale, sisteme cuantice, calcul ADN etc.

Un microcalculator IBM sau o clonă PC este formată din trei părți: unitatea sistem (eng. *system unit*), tastatura (eng. *keyboard*) și ecranul (eng. *display screen*).

Unitatea sistem este construită modular. Partea cea mai importantă este placa de bază (eng. *mother board* sau *main board*). Ea conține circuitele electronice cele mai importante: micropresorul, ceasul (eng. *clock*), coprocesorul matematic (la generațiile mai vechi), memoria RAM (eng. *Random Access Memory*) și memoria ROM (eng. *Read Only Memory*). În plus, unitatea sistem mai conține: sursa de alimentare, ventilatoare și unitătile de discuri. Diferitele placete pentru cuplarea unor dispozitive de intrare-iesire (imprimantă, disc, modem etc.) se cupleză la placa de bază folosind conectorii la magistrală. Magistrala (eng. *bus*) este un canal comun de comunicație între placetele calculatorului prin care circulă semnalele de dialog între componente. Placetele pot fi introduse optional în sloturi (conectori), în funcție de dorința utilizatorului. Cele mai importante placete sunt: adaptorul video (eng. *display screen adapter*), adaptor sau cuploul de disc, placetele de memorie, placetele de comunicație (pentru imprimantă, modem etc.).

Pentru microcalculatoare compatibile IBM-PC se utilizează mai multe tronsoane de magistrală: ISA (eng. *Industry Standard Architecture*), MCA (eng. *Micro Channel Architecture*), EISA (eng. *Extended Industry Standard Architecture*) și PCI (eng. *Peripheral Component Interconnect*).

Prin intermediul magistralei se asigură o arhitectură deschisă, astfel că utilizatorii pot extinde sistemul prin inserarea de placete, cum a fost precizat mai sus. Totuși numărul sloturilor este limitat. Conectarea unui număr suplimentar de dispozitive se poate realiza prin intermediul unui adaptor SCSI (eng. *Small Computer System Interface* – se pronunță “scuzzy”). Se poate conecta până la 7 dispozitive: imprimante, discuri rigide, unități CD-ROM / CD-RW, etc.

Minicalculatoarele sunt sisteme de calcul cu dimensiune și performanțe de prelucrare situate între cele ale microcalculatoarelor și sistemelor *mainframe*. Ele sunt utilizate de companiile medii sau departamentele companiilor mari pentru monitorizarea proceselor de fabricație, gestiunea economică sau cercetare.

Sistemele *mainframe* sunt calculatoare de dimensiune mare care necesită condiții speciale de funcționare (de exemplu, aer condiționat), dar au putere mare de calcul și pot stoca cantități "uriașe" de date. Sunt utilizate de organizațiile mari – companii multinaționale, agenții guvernamentale, bănci, universități etc. – pentru a prelucra un număr foarte mare de tranzacții pe unitatea de timp.

Supercalculatoarele sunt cele mai puternice sisteme de calcul. Ele sunt utilizate în cercetare de organizații puternice pentru explorarea resurselor, simulări, predicții etc.

I.3. Procesor. Caracteristici. Set de instrucțiuni

Una dintre componentele principale ale oricărui sistem de calcul o reprezintă procesorul. Procesorul actualelor sisteme poate fi un microprocesor sau un ansamblu integrat de microprocesoare. Orice procesor conține patru blocuri funcționale de bază:

- unitatea de comandă și control (UCC),
- unitatea aritmético-logică (UAL),
- registrele procesorului,
- unitatea de interfață cu celelalte componente (UI).

Procesoarele performante utilizează structuri de date avansate precum stivele. Acestea sunt utile pentru salvarea contextului unei activități înainte de întreruperea acesteia. Primele trei componente formează unitatea de executare.

UCC comandă, coordonează și controlează întreaga activitate de prelucrare la nivelul componentelor calculatorului. Ea va executa instrucțiunile unui program.

UAL realizează prelucrarea datelor cerută prin instrucțiuni: operații aritmetice, logice, de comparare etc.

Registrele reprezintă o memorie foarte rapidă a procesorului în care se păstrează codul instrucțiunilor, datele de prelucrat, rezultatele prelucrărilor etc. Cele mai importante registre ale unui procesor sunt:

- *registrul acumulator*,
- *registrul numărător de adrese al programului*,
- *registrul indicatorilor de condiții* (valoare negativă, pozitivă sau nulă, transport în urma executării operațiilor de calcul etc.),
- *registrul de instrucțiuni și registrul de adresare a memoriei*.

În general, registrul acumulator păstrează unul dintre operanții unei instrucțiuni de calcul, fiind totodată și destinația rezultatului operației. Registrul numărător de adrese al programului sau registrul contor-program, arată adresa, în memoria internă, unde se află stocată următoarea instrucțiune de executat. Indicatorii de condiție sunt poziționați automat în urma efectuării anumitor operații. Registrul de instrucțiuni memorează instrucțiunea ce se execută. Conținutul acestui registru este analizat pentru a se determina operația de executat, locul unde se află

stocați operanții precum și locul unde va fi stocat rezultatul, dacă instrucțiunea este una de calcul, respectiv adresa unde se va face un salt în program sau adresa unei zone de memorie unde/de unde se va stoca/citi o anumită dată, în alte situații. Registrul de adresare a memoriei păstrează adresa curentă folosită pentru efectuarea accesului la memorie. De obicei, adresa efectivă se obține în urma unui calcul de adresă.

UI asigură legătura dintre procesor și celelalte componente ale calculatorului îndeplinind funcția de transfer de date de la/spre procesor. Comunicarea microprocesorului cu celelalte componente: adaptorul video, adaptorul de disc etc. se face prin intermediul porturilor (puncte de intrare în microprocesor). Acestea pot fi porturi de intrare (vin date de la componente) respectiv porturi de ieșire (pornesc date spre componente). În practică, un port este identificat printr-un număr (unic).

Deoarece un sistem de calcul execută mai multe activități, acestea pot avea nevoie de controlul procesorului. Rezultă necesitatea întreruperii unei activități pentru a trece la o altă activitate. Aceste comutări sunt determinate fie prin hardware, fie prin software. Înteruperea hardware este declanșată la apariția unui semnal de înterrupere prin care procesorului își cere să analizeze un anumit eveniment.

Performanțele procesorului pot fi exprimate prin: *durata ciclului procesorului, lungimea cuvântului, repertoriul de instrucțiuni, numărul adreselor dintr-o instrucțiune, durata executării instrucțiunilor* etc.

Durata ciclului procesorului reprezintă intervalul de timp în care se efectuează un transfer între două registre ale procesorului.

Lungimea cuvântului poate fi de 8 biți, 16 biți, 32 biți, 64 biți etc. în funcție de tipul procesorului.

Repertoriul de instrucțiuni conține cel puțin următoarele grupe de operații (mnemonicele instrucțiunilor au caracter orientativ):

- *instrucțiuni generale* (MOV - mutare de informație, PUSH - punere de informații în memoria organizată ca o stivă, POP - aducerea informației din memoria stivă etc.);
- *instrucțiuni de intrare/ieșire* (IN - depunerea în registrul acumulator a informației stocate în registrul de intrare/ieșire (portul de date), OUT - scrierea în portul de date a informației aflate în registrul acumulator);
- *instrucțiuni aritmetice*: adunare (ADD, ADC, INC etc.), scădere (SUB, DEC, NEG, CMP etc.), înmulțire (MUL, IMUL etc.), împărțire (DIV, IDIV etc.);
- *instrucțiuni de manipulare a șirurilor de biți*: operații logice (NOT, AND, OR, XOR, TEST), deplasare (SHL, SAL, SHR, SAR), rotire (ROL, ROR, RCL, RCR);
- *instrucțiuni de transfer*: salt necondiționat (CALL, RET, JMP), salt condiționat
 - prin testarea indicatorilor de condiții (JC, JNC, JE, JG, JL etc.), cicluri (LOOP etc.), întreruperi (INT, IRET etc.);
- *instrucțiuni de sincronizare externă*: HLT, WAIT, NOP etc.

Durata executării unei instrucțiuni reprezintă timpul necesar desfășurării fazei de citire-interpretare și a fazei de execuție a acelei instrucțiuni.

Pentru microcalculatoare, procesorul este reprezentat de un singur circuit integrat (eng. *silicon chip*), nunit microprocesor (eng. *microprocessor* - "microscopic processor"). Microcalculatoarele prelucră (procesează) date și instrucțiuni în milionimi de secundă, sau *microsecunde*, supercalculatoarele realizează prelucrări în nanosecunde și chiar picosecunde, ele fiind de peste un milion de ori mai rapide decât microcalculatoarele.

În acest moment, există două categorii de microprocesoare: CISC (eng. *Complex Instruction Set Computer*) și RISC (eng. *Reduced Instruction Set Computer*). Cele mai răspândite microprocesoare sunt cele CISC (având ca reprezentanți microprocesoarele INTEL : *X86, Pentium*). Microprocesoarele RISC implementează mai puține instrucțiuni, sunt mai simplu de proiectat și mai ieftine. Căteva exemple de microprocesoare RISC sunt: *PowerPC* – dezvoltat de Motorola, IBM și Apple; *Alpha* și *MIPS R400* – dezvoltate de Digital Equipment Corporation. O abordare specială o reprezintă arhitectura CISC bazată pe tehnici de tip RISC care duce la o viteză de executare a aplicațiilor CISC comparabilă cu cea a sistemelor RISC. Este cazul soluțiilor AMD și Cyrix.

I.4. Memoriile

Stocarea informațiilor în sistemul de calcul se realizează prin intermediul memoriei. Memoria este spațiul de lucru primar al oricărui sistem de calcul. În funcție de locul ocupat, distingem: *memoria centrală* (numită și *memoria principală* sau *internă*) și *memoria secundară* (numită și *auxiliară* sau *secundară*). În memoria centrală sunt stocate programele și informațiile utilizate de ele în timpul execuției lor de către procesor. Memoria secundară păstrează cantități mari de date și programe folosite frecvent și încărcabile rapid în memoria centrală. Memoria unui sistem de calcul este caracterizată prin: *capacitate, timp de acces, viteză de transfer, cost, mod de accesare a informației stocate* etc.

Capacitatea memoriei este definită prin numărul de unități de informație (caracter, cuvinte) disponibile pentru stocarea informației. În general, capacitatea memoriei se exprimă în MB (mega octeți, eng. mega bytes): $1 \text{ MB} = 1024 \text{ KB} = 1024 * 1024 * 8 \text{ biți}$). Localizarea zonelor de memorie (internă sau externă) se realizează prin intermediul adresei. Prima locație a memoriei centrale are adresa 0 (zero). Dacă unitatea de stocare este octetul, atunci adresa 10 (scrisă zecimal) reprezintă locația a unsprezeceea.

Timpul de acces exprimă durata intervalului în care poate fi obținută informația stocată în memorie. Viteză de transfer se exprimă prin numărul de unități de informație transferate de memorie în unitatea de timp.

În funcție de natura accesului la informația ce o înmagazinează, memoria centrală a unui calculator poate fi cu acces numai în citire și respectiv, cu acces în scriere și citire. Memoria de tip "numai citirea este permisă" (ROM- eng. "*Read Only Memory*") se mai numește și *memorie permanentă* sau *nevolatilă*, deoarece programele și datele ce au fost înscrise în ea sunt fixate o dată pentru totdeauna. În

general, în această memorie este depozitat *firmware*-ul³ (secvență de instrucțiuni cu destinație specială). Memoria ce permite acces de tip "citire-scriere" se mai numește memorie cu acces aleator (RAM- eng. *Random Access Memory*) - deși, tot cu acces aleator este și memoria ROM - și este de tip *volatile* (trebuie să fie alimentată electric pentru a reține date). Din punct de vedere tehnologic deosebim două tipuri de memorie RAM: DRAM (RAM dinamic) și SRAM (RAM static). Atributul dinamic specific necesită unui interval de timp foarte mic între momentele de reîmprospătare (eng. *refresh*), reîmprospătarea realizându-se de sute de ori pe secundă pentru a se reține datele stocate în celulele de memorie. DRAM-ul este preferat pentru memoria principală a sistemelor, în timp ce SRAM-ul, care nu necesită o reîmprospătare foarte deasă, este utilizat în primul rând la implementarea memoriei *cache*⁴. Rolul memoriei cache constă în memorarea datelor și instrucțiunilor solicitate frecvent de către procesor. Memoria cache este primul loc unde procesorul cauță datele de care are nevoie. Numai dacă acestea nu se află în memoria cache ele vor fi căutate în memoria principală.

Există mai multe tipuri de module DRAM utilizate în sistemele de calcul moderne: FPM (eng. *Fast Page Mode*), EDO (eng. *Enhanced Data Out*), SDRAM (eng. *Synchronous DRAM*) și.a. Ultimul tip amintit, face ca procesul de interogare a memoriei principale să fie foarte eficient datorită sincronizării modulului de memorie cu semnalul de ceas al procesorului. Circuitele de stocare de tip ROM se încadrează în următoarele clase: PROM (eng. *Programmable Read-Only Memory*) – pentru înregistrarea codului cu ajutorul unui echipament special, odată ce este scris nu se mai poate schimba; EPROM (eng. *Erasable Programmable Read-Only Memory*) – circuit de stocare de tip ROM care poate fi șters cu ajutorul unui mediu în ultraviolet, iar apoi poate fi rescris.

I.5. Dispozitive periferice

Totalitatea echipamentelor unui sistem de calcul diferite de unitatea centrală și memoria internă formează mulțimea echipamentelor periferice. Din această categorie fac parte unitățile de memorie externă, echipamentele de intrare, echipamentele de ieșire și echipamentele de intrare/ieșire.

³ *Firmware* este un cuvânt care inițial a fost folosit pentru a desemna microprogramele cu ajutorul cărora se realiza unitatea de comandă și control a unui procesor. Astăzi desemnează și secvențele de cod (în limbajul procesorului) ce implementează interpretoare, nuclee de intrare- ieșire etc. De asemenea, această componentă este utilă în implementarea standardului PnP (eng. *Plug and Play*) util în reconfigurarea automată a sistemelor de calcul.

⁴ Memoria de tip *cache* poate apartine atât procesorului (fiind integrată acestuia), dar și spațiului RAM. De aceea, recent, se utilizează organizarea stratificată – pe niveluri - a memoriei cache.

I.5.1. *Discurile magnetice*

Una dintre cele mai importante unități de memorare externă este unitatea de disc magnetic. Memorarea informației pe discul magnetic urmează același principiu fizic cu cel utilizat în înregistrarea casetelor și benzilor audio-video. Deosebirea principală între cele două sisteme de memorare este datorată naturii semnalului de înregistrare folosit: *analogic*, în cazul audio-video și *numeric* (eng. *digital*), în cazul discurilor sistemelor de calcul. Diferența față de benzile magnetice (înregistrate tot numeric) constă în modul de acces. Pe bandă accesul este secvențială, pe disc accesul este direct, după cum se va vedea mai jos.

Din punct de vedere fizic, o unitate de disc magnetic are în compunere unul sau mai multe platane, fiecare platan cu una sau două suprafete de înregistrare, atâtea capete de citire/scriere câte suprafete de înregistrare există, un braț ce permite accesul de la exterior spre centrul platanului al capetelor de citire/scriere, motoare pentru rotirea planelor și deplasarea brațului, precum și un set de circuite specializate (numit controler, cupluri sau adaptor) pentru comanda întregului mecanism.

Discurile magnetice care conțin un singur platan se numesc discuri floppy, iar platanul propriu-zis se numește *dischetă* sau *disc flexibil*. Discurile conținând mai multe platane se numesc *discuri rigide* sau discuri dure (eng. *harddisk*). Atât pentru dischete, cât și pentru discul rigid, întreaga colecție de date formează aşa-numitul *volum*. Suprafețele de înregistrare ale unui disc magnetic sunt împărțite într-un număr fix de cercuri concentrice, fiecare cerc numindu-se *pistă*. Adresa unei piste este definită de o pereche de numere întregi reprezentând numărul curent al suprafeței de înregistrare, respectiv numărul curent al pistei. Suprafețele de înregistrare se numerotează începând de la zero, de sus în jos. Pistele se numerotează crescător de la pista de rază maximă (indice 0) spre pista de rază minimă. Această numerotare se reia pentru fiecare suprafață de înregistrare.

Mulțimea pistelor de pe toate suprafețele de înregistrare identificate prin același număr formează un *cilindru*. Cilindrii se numerotează de la zero, crescător, începând cu cel cu diametrul maxim spre cel cu diametrul minim. Fiecare pistă este împărțită în sectoare, de lungime fixă, de regulă 512 octeți. Fiecare sector este adresat fizic de un triplet de numere întregi reprezentând numărul cilindrului, numărul suprafeței și numărul sectorului. Numerotarea sectoarelor pe cilindru începe de la unu. Pe fiecare cilindru, numărul de sectoare este același.

Din punct de vedere logic, pistele sunt numerotate începând de la zero pe suprafața de înregistrare zero și continuând crescător până la epuizarea suprafețelor și pistelor. De asemenea, sectoarele sunt numerotate începând de la zero (cilindrul zero, suprafața zero), crescător până la epuizarea sectoarelor suprafețelor cilindrilor. Adresa fiecărui sector este înscrisă la începutul acestuia într-un antet ce mai conține și o secvență de octeți de sincronizare utilizată de adaptor (cupluri) pentru identificarea unei adrese disc. Operația de scriere a adresei și a celorlalte elemente ale antetului sectoarelor se numește *formatare fizică* a discului. În cazul sistemului de operare MS-DOS, formatarea fizică a dischetelor este realizată de

comanda **FORMAT**. Pentru discul rigid, formatarea fizică este realizată, în majoritatea cazurilor, de către fabricantul acestuia.

Cele de mai sus arată că discul magnetic este o memorie accesabilă prin adresă, deci datele se obțin prin acces direct.

Performanțele unui disc magnetic depind de următorii factori: timpul de poziționare, numărul de rotații pe minut, rata de transfer a discului etc.

Discurile flexibile au două suprafete pe care se poate scrie informația și pot fi protejate la scriere prin fanta de protecție (în poziția liber). Cuplarea unităților de disc se poate realiza *intern* (prin intermediul unei interfețe standard: *IDE*, *SCSI* etc.) sau *extern* (prin intermediul interfeței paralele).

Din punct de vedere constructiv, discurile rigide sunt de tip *intern* (capacitate fixă, suportul de memorare nu poate fi demontat din unitatea de disc), *cartridge* (capacitate variabilă – suportul poate fi evacuat precum o casetă din cititorul/inregistratorul de casete (casetofon sau videocasetofon), respectiv *pachet* de discuri (care se montează în unitățile speciale de citire-scriere ale mini și supercalculatoarelor).

I.5.2. *Discurile optice*

Cele mai recente unități de memorie externă sunt unitățile de disc optic sau magneto-optic. Înscrierea optică a informației este un proces care folosește un fascicul de lumină (o rază laser) fie pentru citirea și/sau înregistrarea datelor, fie pentru a ajuta la realizarea acestor operații pe un suport sensibil optic. Citirea datelor se bazează pe evidențierea unor modificări survenite în fascicul de lumină reflectată de suport. Operația de scriere folosește un fascicul laser pentru modificarea (sau pentru a facilita modificarea) unui material sensibil la lumină amplasat pe suport. Din punct de vedere tehnologic distingem următoarele tipuri de discuri optice:

- *discurile CD-ROM* (eng. *Compact Disc – Read-Only Memory*): Sunt discuri compacte preînregistrate pe care informația nu poate fi actualizată. Conținutul unui astfel de disc este înscris de către producătorul său și poate fi doar citit de către utilizator.
- *discurile CD-R* (eng. *CD-Recordable*): Sunt discuri compacte ce pot fi înregistrate de către posesor, dacă acesta dispune de un dispozitiv periferic special și programele de bază aferente. După înregistrare, fostul CD-R devine un CD-ROM pe care îl poate citi orice unitate de citire CD-ROM. Discurile CD-R se mai numesc discuri WORM (eng. *Write Once, Read Many*).
- *discurile CD-E*: sunt discuri compacte de pe care informația poate fi ștersă. Aceste discuri suportă un număr limitat de cicluri de citire/scriere.
- *discuri MO (magneto-optice)*: Sunt discuri la care scrierea se realizează magnetic, iar citirea optică. Aceste discuri pot fi scrise de câte ori este necesar, atâtă vreme cât suportul nu este deteriorat. Ele fac parte din clasa, mai largă, a discurilor RW (eng. *ReWritable optical discs*).

Caracteristicile principale ale unui disc optic sunt: dimensiunea (5,25" sau 3,5"), capacitatea de stocare, numărul de cicluri de citire/scriere (pentru CD-E) etc.

Discurile compacte înregistrabile tind să devină cel mai convenabil suport pentru salvarea și transportul datelor. Ele pot stoca un volum mare de date (sute de MB), iar costul este din ce în ce mai mic. Astfel, ele tind să înlocuiască dischetele, benzile și casetele magnetice în multe aplicații de transport și arhivare a datelor.

I.5.3. *Terminalul*

Deoarece unitățile de memorare externă, de obicei, permit atât scrierea cât și citirea informației, aceste dispozitive periferice pot fi considerate și ca dispozitive de intrare- ieșire. Un alt dispozitiv periferic de intrare- ieșire este terminalul. Prin intermediul acestuia se asigură comunicația utilizator - sistem de calcul. Prin intermediul acestuia utilizatorul poate conversa cu calculatorul.

La un sistem de calcul pot fi cuplate unul sau mai multe terminale. Din punct de vedere al modului de cuplare la sistemul de calcul distingem două tipuri de terminale: *terminale seriale* și *terminale "memorie"*.

Terminalul serial este cuplat la calculator prin intermediul interfeței de comunicație standard RS-232 (conector RS-232C). El este compus din *tastatură* (ca dispozitiv de intrare) și un *monitor* (ca dispozitiv de ieșire). Pentru a transmite un caracter unui astfel de terminal, calculatorul (prin circuitele sale de interfață) transmite: *un bit de start, biți ce compun caracterul și unul sau doi biți de stop*. Atât terminalul cât și calculatorul dispun de circuite hardware specializate în conversia sir de caractere - sir de biți și invers.

Interfața unui terminal "memorie" cu calculatorul este asigurată prin intermediul unei memorii RAM cu destinație specială numită *memorie video*. Această memorie este adresabilă ca și memoria principală. Controlul memoriei video și generarea semnalului video pentru monitor sunt realizate de un set de circuite care formează *adaptorul video*. La aceste terminale, tastatura este separată de monitor. Ea este cuplată printr-o interfață paralelă (mai mulți biți simultan), dar există și tastaturi cuplate serial. Pentru sistemele de calcul ce utilizează terminale "memorie", tastatura este considerată un dispozitiv periferic separat, din clasa dispozitivelor periferice de intrare.

Prin intermediul tastaturii utilizatorul poate transmite comenzi calculatorului. Comenzile se introduc sub forma unui sir de caractere (litere, cifre, semne speciale, caractere de control). Fiecare caracter se generează prin acționarea uneia sau mai multor taste (butoane; eng. *keys*). Acționarea unei taste sau combinații de taste are ca efect închiderea unui circuit electronic prin care se generează un cod unic, care este codul ASCII (sau Unicode) al caracterului respectiv. Orice tastatură modernă are patru blocuri de taste:

1. *Tastatura mașinii de scris*. Sunt disponibile butoane pentru introducerea următoarelor date simple: cifre, literele mari și mici, semne speciale, bara de spațiu, return de car (CR - *Carriage Return*), salt la linie nouă (LF - *Line Feed*), saltul cursorului cu un număr de coloane (*Tab*), întreruperea unei activități (*Esc- Escape*), tipărirea imaginii ecranului sau memorarea acesteia (*Print Screen*), suspendarea temporară a executării unui program (*Pause/Break*).

2. Tastatura de editare. Editarea unui text constă în scrierea textului și corectarea acestuia. Textul este afișat pe ecran. Acolo apare și o bară luminoasă sau un indicator special (cursor⁵) care arată poziția în care va fi inserat următorul caracter. Sunt disponibile următoarele butoane: tastele săgeți (pentru deplasarea cursorului), tastele *Page Up* și *Page Down* care comandă deplasarea cursorului în sus / jos cu un anumit număr de rânduri, tastele *Home* și *End* care comandă deplasarea cursorului la începutul / sfârșitul textului, tasta *Insert* pentru alegerea tipului de corecție (cu suprascriere sau cu inserare), tastele pentru ștergere (*Delete* șterge caracterul din poziția cursorului, *Backspace* șterge caracterul de la stânga cursorului).
3. Tastatura numerică conține tastele pentru cifre și operațiile aritmetice: + (adunare), - (scădere), * (înmulțire), / (împărțire) și . (punct) ca separator între partea întreagă și partea zecimală a unui număr (conform sistemului englezesc de scriere).
4. Tastele funcționale: cele 12 butoane F₁, F₂, ..., F₁₂ cărora li se pot asocia diferite acțiuni de către programatorul de aplicații. Un utilizator, înainte de utilizarea tastelor funcționale, în cadrul unei noi aplicații, trebuie să inventarieze lista asocierilor. Aplicații diferite fac asocieri diferite, pentru aceeași tastă. Chiar, în cadrul aceleiași aplicații, asocierea se poate schimba de la un nivel funcțional la altul.

Unele taste sunt de tip "cald" (eng. *hotkeys*), iar altele de tip "rece" (eng. *coldkeys*). Tastele reci nu generează cod către calculator ci se folosesc împreună cu tastele calde pentru a realiza combinații. Tastele reci sunt: *Shift*, *Ctrl*, *Alt*. De exemplu, majoritatea programelor aplicative folosesc combinațiile: *Ctrl+N* (pentru lansarea unui nou proiect: program, document, imagine, desen tehnic etc; eng. New), *Ctrl+O* (încarcă un proiect existent pentru actualizare, tipărire etc; eng. Open), *Ctrl+S* (înregistrează pe un suport de memorare externă proiectul curent; eng. Save), *Ctrl+P* (imprimă "imaginea" proiectului curent folosind o imprimantă sau trimită proiectul prin intermediul unui adaptor de tip fax; eng. Print), *Ctrl+X* (mută o parte a unui proiect într-o zonă a memoriei RAM, numită *clipboard*; eng. Cut), *Ctrl+C* (copiază o parte a unui proiect în *clipboard*; eng. Copy), *Ctrl+V* (copiază conținutul zonei *clipboard* în proiectul curent, în locul specificat de utilizator; eng. Paste) etc.

O tastatură are și taste comutator (cu două stări): *CapsLock* (asigură comutarea între starea care generează litere mici și starea care generează litere mari), *NumLock* (comută între starea numerică și starea de editare pentru blocul tastelor numerice), *Insert* (comută între corecție prin inserare și corecție cu suprascriere).

Monitorul oricărui terminal (compus din ecran și circuite de generarea imaginii) poate lucra în două moduri: modul *text* și modul *grafic*.

În modul text ecranul este împărțit în rânduri (eng. *rows*) și coloane (eng. *columns*). Numărul de rânduri și numărul de coloane este dat de modul de lucru permis de monitor. De obicei sunt 25 de rânduri și 80 de coloane. La intersecția

⁵ A nu se confunda cu sensul precizat în DEX'1996.

unei linii cu o coloană se generează un caracter printr-o matrice de puncte luminoase. Pentru fiecare poziție de afișare, se vor păstra (din motive de reîmprospătare a imaginii) codul ASCII (Unicode) al caracterului și atributul caracterului (cel care controlează aspectul caracterului afișat și depinde de adaptorul folosit). De exemplu, pentru calculatoare personale, codul caracterului este stocat folosind 8 biți, iar atributul folosind alți 8 biți. Atributul pentru afișarea color este format din trei elemente: culoarea penișei (eng. *foreground*), culoare hârtiei (eng. *background*) și elementul de control al clipirii (eng. *blink*). Culoarea este specificată cu ajutorul a trei componente fundamentale: R-roșu (eng. *Red*), G-verde (eng. *Green*) și B-albastru (eng. *Blue*).

În modul grafic ecranul este o suprafață de puncte luminoase numite *pixeli* (elemente de imagine; eng. *picture element*). Fiecare pixel este caracterizat prin codul culorii. Următoarele elemente characterizează un anumit mod grafic:

- *Rezoluția* - numărul de puncte de pe ecran;
- *Definiția* - distanța dintre două puncte pe ecran;
- *Numărul de culori*.

Toate aceste elemente depind de modul grafic suportat de monitor. De exemplu modul VGA (eng. *Video Graphics Array*) asigură o rezoluție de 640 x 480 puncte și 16 culori, iar modul SVGA (eng. *Super VGA*) standard asigură rezoluția de 800 x 600 pixeli în 256 culori. Modul XGA (eng. *eXtended Graphic Array*) afișează mai mult de 18 milioane de culori pentru o rezoluție de până la 1024 x 768 de pixeli.

Generarea unor imagini complexe, la o viteză de prelucrare mare, fără aportul procesorului sistemului de calcul, este asigurată astăzi de coprocesoarele grafice (numite și acceleratoare) care realizează în mod independent: trasarea liniilor, generarea suprafețelor definite prin contur, desenarea umbrelor, deplasarea textului, deplasarea blocurilor de imagine etc.

Monitoarele sunt de tip desktop (la locul de muncă sau acasă) și portabile. Monitoarele desktop, numite și sisteme CRT (eng. *Cathode-Ray Tubes*) sunt similare ca dimensiune și tehnologie cu receptoarele emisiunilor de televiziune. Sunt de tip întrețesut (eng. *interlaced*) – cu realizarea imaginii pe rândurile impare și apoi pe cele pare și, de tip neîntrețesut (noninterlaced). Întrețeserea provoacă pâlpâirea, și deci oboseala ochilor. Monitoarele portabile echipează sistemele de tip laptop, notebook, subnotebook și PDA.

I.5.4. *Dispozitive pentru introducerea informației grafice*

În categoria dispozitivelor periferice de intrare sunt incluse și echipamentele: *mouse*⁶, *joystick*, *trackball*, tabletă grafică sau digitizor (eng. *digitizer*), creion optic (eng. *light pen*) etc. Primele trei dispozitive de interacțiuie controlează deplasarea unui cursor pe ecranul unui sistem de calcul. Diferă numai

⁶ Acest dispozitiv este denumit maus (plural: mausuri) în DEX'1996: "dispozitiv actionat manual, conectat la un computer, a cărui deplasare pe o suprafață antrenează deplasarea cursorului pe ecran."

constructiv. Mausul dispune de butoane a căror apăsare este interpretată de programele sistemului de calcul care generează o secvență de operații specifică locului cursorului, butonului apăsat și funcțiilor programului în executare. Creionul optic este un dispozitiv de selecție și se utilizează numai în combinație cu terminale speciale, pentru aplicații speciale (ex. proiectare grafică, pictură asistată de calculator etc.). Tableta grafică este un digitizor ce poate fi folosit fie pentru selecție, fie pentru introducere de date în aplicații de proiectare inclusiv pentru arhitectură, sisteme informatiche geografice etc.

Mous-ul se poate deplasa pe o masă reală (eng. *pad*) și va antrena deplasarea unui cursor pe ecran. Mausul are mai multe butoane utile în efectuarea următoarelor operații:

1. indicare (eng. *point*) - cursorul mausului este deplasat pentru a indica un anumit punct de pe ecran (deci reprezentarea unui anumit obiect);
2. clic (eng. *click*) - se acționează, foarte scurt, un buton al mausului. Codul ce controlează funcționarea mausului va trata evenimentul apărut;
3. clic dublu (eng. *double click*) - se acționează, foarte scurt, de două ori, un buton al mausului;
4. tragere (eng. *drag*) - se asigură deplasarea mausului pe masa reală, acesta având un buton apăsat continuu.

Tabletele grafice se pot clasifica pe baza a două criterii:

- a) după dimensiunea suprafeței active: A4, A3, A0;
- b) după precizie și acuratețe: pentru digitizare de planuri și pentru meniuri.

Ele pot fi echipate cu un *stylus* sau un *puck* cu 4-16 butoane programabile.

Un alt periferic de intrare, cu aplicații în introducerea imaginilor, este scannerul (eng. *scanner*). După citirea imaginii, aceasta poate fi prelucrată: mărită, micșorată, rotită, colorată, suprapusă cu alte imagini și analizată folosind diferite metode. Principiul fundamental al funcționării scannerului îl reprezintă modificarea intensității unui fascicul luminos la întâlnirea unei suprafețe de o culare oarecare. Scanarea unui document se desfășoară în două faze. Un fascicul luminos, în prima fază, baleiază (scanează) documentul linie cu linie, iar fascicul luminos care se reflectă este direcționat (cu ajutorul unui sistem de oglinzi și lentile) spre o mulțime de celule fotosensibile CCD (eng. *Charge-Coupled Device*). În etapa următoare, CCD-ul transformă semnalele luminoase recepționate în semnale electrice care după o conversie analog-digital sunt trimise programului care va salva imaginea. Scanările color obțin trei versiuni ale documentului de scanat: una de culoare roșie (eng. *red*), una de culoare verde (eng. *green*) și una de culoare albastră (eng. *blue*) care contribuie la imaginea finală. Observăm că ceea ce se introduce nu este un punct ci o suprafață de puncte. Caracteristicile unui scanner sunt: a) rezoluția optică- numărul de puncte pe unitatea de suprafață pe care le poate citi (eng. *dots per inch*); b) numărul de culori și c) viteza de scanare (explorare).

Alte scanere sunt specializate: cititorul de bare (eng. *Bar Code Reader*), cititorul de taloane (eng. *Badge Reader*), cititorul de text (eng. *Document Scanner* sau *OCR Scanner*), cititoare cu cerneală magnetică a înscrисurilor bancare (eng. *Magnetic Character Ink Reader*) etc.

Transmiterea/Recepționarea documentelor la/de la distanță se poate realiza, pe linie telefonică, folosind un adaptor special (modem) și un fax (eng. facsimile transmission machines). În prezent cele două componente sunt integrate pe o placă numită *internal fax-modem* sau într-un dispozitiv extern care se cuplăază la un sistem de calcul prin portul serial.

I.5.5. Echipamente de ieșire

Din categoria echipamentelor de ieșire ne vom referi la următoarele: imprimanta, plotterul și fotoplotterul. Plotterele au fost primele dispozitive periferice care au oferit sistemelor de calcul posibilitatea de a produce ieșiri în formă grafică. Plotterele pot fi cu penită, cu jet de cerneală, de tip termic și de tip electrostatic.

Plotterele au în componență procesoare grafice proprii. De exemplu, plotterele cu penită recunosc primitive grafice precum: linie, poligon, arc de cerc, text etc. Dacă suportul de informație este filmul fotografic atunci dispozitivul asemănător plotterului, dar instalat în codiții specifice se numește fotoplotter.

Imprimantele sunt dispozitive de afișare alfanumerică sau grafică. Ele pot fi: cu ace, cu jet de cerneală, cu transfer termic sau pe bază de laser.

Sistemele multimedia acceptă și intrare/ieșire sonoră disponând de dispozitive专特 pentru analiza/sinteză vocală. Conversia vocală unei persoane în cod numeric se realizează în scopul recunoașterii vorbirii. Sunt disponibile sisteme integrate pentru recunoașterea vorbirii (la nivel discret și la nivel continuu). Acestea sunt sisteme speciale care realizează și traducerea dintr-o limbă în alta, a mesajului vorbit.

I.6. Funcționarea unui sistem de calcul

Am văzut că un sistem de calcul este un dispozitiv automat în care datele reprezentate în binar sunt prelucrate pe baza unui program ce indică o succesiune determinată de operații. Datele inițiale de prelucrat și programul constituie din instrucțiuni se introduc în sistemul de calcul prin intermediul unor dispozitive periferice de intrare. Prin intermediul unor canale de comunicație, datele și instrucțiunile sunt transferate în memoria internă sub formă binară, în locații identificabile prin adresele la care au fost memorate (și nu prin conținutul acestora). Apoi fiecare instrucțiune este trimisă la UCC care interpretează conținutul acesta și trimită comenzi către:

1. memorie - prin care se solicită ca anumite date, localizate prin adresele la care sunt memorate, să fie transferate către UAL pentru execuția anumitor operații; după realizarea operației se va indica adresa din memorie unde se va depune rezultatul operației efectuate de UAL;
2. UAL - își va solicita execuția operației indicate de instrucțiune;
3. canalele de intrare- ieșire - pentru preluarea altor date și instrucțiuni de la dispozitivele de intrare- ieșire, respectiv pentru începerea transferului rezultatelor din memorie către dispozitivele periferice de ieșire.

După terminarea execuției operațiilor solicitate, rezultatele memorate la anumite adrese din memorie sunt transferate către dispozitivele de ieșire pentru vizualizarea acestora.

I.7. Tipuri de sisteme de calcul

Sistemele de calcul pot fi de tip numeric (digitale), analogic și de tip hibrid. Calculatoarele numerice, sunt cele care primesc, prelucră și transmit date/informări codificate binar. Ele fac obiectul acestei lucrări. Sistemele de calcul analogice sunt echipamente alcătuite din amplificatoare operaționale și circuite numerice independente, interconectate în vederea realizării unor operații de tip continuu: integratoare, multiplicatoare etc. Mărurile corespunzătoare condițiilor inițiale se introduc sub forma unor tensiuni electrice. Acestea sunt prelucrate și se obțin noi tensiuni electrice ce vor fi vizualizate cu ajutorul unor dispozitive speciale. Sistemele hibride sunt rezultatul cuplării unor sisteme numerice cu sisteme analogice. Comunicarea între cele două tipuri de sisteme se realizează prin intermediul unor dispozitive de conversie: analogic-digital; digital-analogic. Sistemele analogice nu fac obiectul acestei lucrări.

O clasificare interesantă a sistemelor digitale a fost propusă de Flynn (1972) și cuprinde atât sistemele de calcul cu o singură unitate centrală, cât și pe cele cu mai multe unități centrale (figura 1.1). Clasificarea lui Flynn consideră ca esențiale două caracteristici: mărimea fluxului instrucțiunilor și mărimea fluxului datelor.

Un sistem de calcul cu flux unic de instrucțiuni și flux unic de date este numit sistem de calcul SISD (eng. *Single Instruction Single Data Stream*). Toate sistemele de calcul tradiționale (cu o singură unitate centrală) sunt mașini SISD. Această categorie include sisteme de calcul, de la microcalculatoare personale până la calculatoarele multiutilizator de mare putere (eng. *mainframe*): IBM/704, IBM/7040, IBM 360/40, IBM 370/135, PDP, FELIX C, microcalculatoare IBM PC etc. Următoarea categorie o reprezintă sistemele de calcul cu flux simplu de instrucțiuni, dar cu flux multiplu de date, numite sisteme SIMD (eng. *Single Instruction Multiple Data Stream*). Aceste sisteme se bazează tot pe o singură unitate centrală cu o singură unitate de comandă, dar cu N elemente de prelucrare (UAL) și N module de memorie atașate acestora, toate interconectate ($N \geq 2$). Unitatea de comandă emite instrucțiuni sub formă de flux unic spre toate elementele de prelucrare, în mod simultan. La un moment dat, toate elementele de prelucrare active execută aceeași instrucțiune, numai asupra datelor situate în propriul modul de memorie, deci flux multiplu de date. Din această clasă fac parte unele *supercalculatoare* (de exemplu, ILLIAC-IV cu 64 UAL și pentru fiecare UAL fiind disponibili 8KB de memorie).

Sistemele SIMD sunt, la rândul lor, de mai multe categorii:

- a) matriceale - prelucră datele în mod paralel și le accesează prin *adrese* în loc de *index* și *valoare*;

- b) cu memorie asociativă - operează asupra datelor accesate *asociativ* (prin conținut). În loc de adresă, specificarea datelor se face prin valoare, cum ar fi: "mai mare decât", "mai mic decât", "între limitele", "egal cu" etc.;
- c) matriceal-asociative - sunt sisteme de tip asociativ ce operează asupra tablourilor multidimensionale (matrice și masive de date);
- d) ortogonale - fiecare element procesor corespunde la un cuvânt (32 biți) de memorie și, astfel, biții de același rang ai tuturor cuvintelor pot fi prelucrați în paralel. Acest procedeu mai este numit *procesare serială pe bit și paralelă pe cuvânt*.

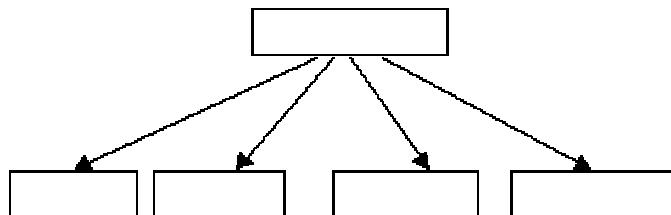


Figura I.1. Tipuri de sisteme de calcul (Flynn)

Clasa sistemelor cu flux multiplu de instrucțiuni și flux unic de date MISD (eng. *Multiple Instructions Single Data Stream*) include acele structuri specializate ce folosesc mai multe fluxuri de instrucțiuni executate pe același flux de date. Ultima categorie o reprezintă sistemele MIMD (eng. *Multiple Instructions, Multiple Data Stream*), ce reprezintă un grup de calculatoare independente, fiecare cu propriul context (program, date etc.). Multe dintre *supercalculatoare* și *toate sistemele distribuite* intră în această clasă. Sistemele MIMD pot fi divizate în două categorii: *sistemele multiprocesor* (cu memorie comună) și *sisteme multicalculator*. Pe de altă parte, fiecare din aceste clase se poate împărtășii în funcție de modul de interconectare. Există două posibilități de interconectare: *magistrală* (similar televiziunii prin cablu) și *comutație* (similar rețelei telefonice). Se obțin astfel patru clase de sisteme MIMD (figura 1.2): sisteme multiprocesor cu magistrală, sisteme multiprocesor comutate, sisteme multicalculator cu magistrală (*rețele de calculatoare*) și sisteme multicalculator comutate (*sisteme distribuite generale*).

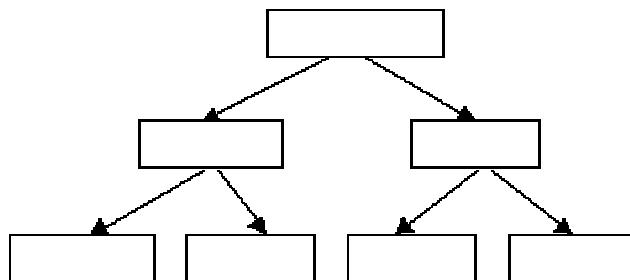


Figura I.2. Tipuri de sisteme MIMD

Calculatoarele dintr-o rețea pot fi de același tip (*rețele omogene*) sau de tipuri diferite (*rețele heterogene*). Rețelele de calculatoare permit folosirea în comun a unor resurse fizice scumpe (imprimante, discuri fixe de mare capacitate etc.) și folosirea în comun a unor date. Datele care se schimbă între sisteme de calcul se numesc *documente electronice*.

În funcție de aria de răspândire a calculatoarelor dintr-o rețea, se disting următoarele tipuri de rețele:

1. *Rețele locale* - LAN (eng. *Local Area Network*): În aceste rețele, aria de răspândire nu depășește 2 km și deservesc o societate comercială. Rețelele locale sunt formate de regulă din calculatoarele aflate într-o clădire sau un grup de clădiri.
2. *Rețele metropolitane* - MAN (eng. *Metropolitan Area Network*): Aceste rețele acoperă suprafața unui oraș.
3. *Rețele globale* - WAN (eng. *Wide Area Network*): Calculatoarele acestor rețele au o arie geografică de răspândire foarte mare (rețele internaționale, "Internet" etc.)

În această lucrare ne vom referi numai la rețelele locale. Rețelele locale, în special bazate pe stații de lucru sau calculatoare personale (PC) prezintă avantaje ce pot fi grupate în trei categorii:

- a) avantaje strategice în mediul de afaceri/educațional;
- b) avantaje operaționale și/sau tactice;
- c) avantaje ale utilizatorului.

Avantajele strategice în mediul de afaceri/educațional includ: facilitarea comunicățiilor în cadrul unei firme/școli, creșterea competitivității firmei/instituției, posibilitatea organizării resurselor în grupuri de lucru cu efect asupra reducerii bugetelor afectate prelucrării datelor. Din punct de vedere operațional și/sau tactic se remarcă: reducerea costurilor per utilizator, creșterea siguranței serviciilor de calcul (prin posibilitatea incluzării serverelor "în oglindă" (eng. *mirror*), îmbunătățirea administrării software-ului, îmbunătățirea integrității datelor (datele de pe server vor fi salvate în mod regulat), îmbunătățirea timpului de răspuns (într-o rețea necongestionată). Un utilizator al unei rețele locale poate avea următoarele avantaje: mediul de calcul poate fi ales de către utilizator, crește repertoriul de aplicații, crește securitatea informației (sistemul de operare în rețea fiind cel care restricționează/permite accesul la datele rețelei), există posibilitatea instruirii on-line (prin intermediul bazelor de date pentru instruire) etc.

Componentele hardware specifice unei rețele locale (în afara stațiilor sau PC-urilor) sunt:

1. *Cabluri* - pot fi coaxiale, fire torsadate (10Base-T), fibre optice etc. Acestea sunt utile pentru realizarea legăturii fizice. Caracteristicile principale ale unui cablu sunt: impedanță, capacitatea, factorul de atenuare, viteza semnalului, caracteristicile de zgomot. Spre deosebire de mediile prin cablu, mediile oferite de telefonia celulară, undele radio terestre, undele radio prin satelit, undele laser, microundele și undelete meteorice permit transmiterea informațiilor fără cablu.

2. *Module de interfață cu rețea* - accesul fizic al unei stații (sau PC) la LAN se face cu ajutorul unui modul de interfață (NIC - eng. Network Interface Card), mai precis o plachetă instalată în PC ce suportă atât funcții de partajare a spațiului fizic cât și funcții de sincronizare.
3. *Tranceiver* - Acestea sunt receptoare-transmițătoare pentru conectare la LAN. Tranceiverul este un echipament ce transmite și recepționează semnal între NIC-ul din PC și mediul fizic utilizat. Dacă sunt utilizate conexiuni Ethernet în T atașate direct la conectorul BNC al unui NIC, atunci nu este necesar un astfel de echipament.
4. *Huburi pentru cablaje* - Sunt echipamente utilizate pentru extinderea zonei de acoperire a rețelei. De exemplu, într-un LAN pe fire torsadate, toate echipamentele sunt conectate într-o configurație de tip stea într-un hub, care este de obicei localizat într-un dulap închis. Un hub suportă de la 8 la câteva sute de stații. Deoarece distanța limită între huburi este de aproximativ 100 de metri, proiectanții utilizează deseori cabluri de interconectare de tip superior (celor 10Base-T) pentru a mări distanța între huburi.
5. *Repetitoare* - Acestea sunt echipamente ce amplifică semnalele pentru a mări distanța fizică pe care poate acționa un LAN.
6. *Punji* (eng. *bridge*) - conectează 2 sau mai multe LAN-uri la nivelul transmiterii pachetelor de date. Ele prelucră datele în funcție de adresa destinatarului și adresa expeditorului.
7. *Rutere* (eng. *router*) - sunt echipamente de dirijare a traficului de date. Ele înțeleg protocolul folosit în transmisia datelor și pot traduce protocoale, putând fi folosite la conectarea a două rețele de calculatoare.
8. *Porți* (eng. *gateway*) - sunt echipamente utilizate pentru conectarea unor rețele de calculatoare care folosesc protocoale diferite.

Din punct de vedere software, accesul la blocurile de instrucțiuni specializate în comanda și controlul unităților de disc, imprimantelor și a altor periferice este asigurat de către sistemul de operare al rețelei (NOS- eng. Network Operating System). Un NOS este un software de rețea ce permite rețelei să suporte capabilități multiproces (eng. *multitasking*) și multiutilizator (vezi capitolul 2). De asemenea un NOS oferă posibilități deosebite pentru partajarea resurselor și pentru comunicare. Cele mai cunoscute NOS-uri sunt: Microsoft LAN Manager, Novell Netware, IBM LAN Server etc.

În afară de NOS-urile tradiționale, recent s-au evidențiat modelul "client-server" și modelul "de la egal la egal" (eng. *peer-to-peer*). Aceste soluții utilizează atât sistemul de operare local al stației cât și NOS-ul.

O arhitectură client-server este un model de calcul în care aplicațiile software sunt distribuite între entitățile din LAN. Clienții solicită informația de la serverul (serverele) din rețea ce stochează datele și/sau programele, partajarea acestora fiind asigurată de NOS. În acest model, un sistem de calcul din rețea este fie server, fie client. Realizarea unor prelucrări cu ajutorul serverului (de la distanță) este realizată prin utilizarea apelurilor RPC (eng. Remote Procedure Call). Alte metode pentru realizarea aplicațiilor client-server sunt: interfața de

programare a aplicațiilor: API (eng. *Application Programming Interface*), serverul de baze de date (eng. *data base*), utilizarea ferestrelor la distanță.

Rețelele bazate pe modelul "de la egal la egal" suportă comunicațiile directe între utilizatori și sunt de dimensiune mică. Fiecare calculator din rețea poate fi, în același timp, și client și server. Cele mai cunoscute sisteme de operare în rețele "de la egal la egal" sunt: Microsoft Windows 9X, Lantastic etc.

II. INTRODUCERE ÎN SISTEME DE OPERARE

II.1 Sistem de operare. Funcții. Clasificare

Un *sistem de operare* este "o colecție organizată de programe de control și serviciu, stocate permanent într-o memorie principală sau auxiliară, specifice tipurilor de echipamente din componența unui sistem de calcul, având ca sarcini: optimizarea utilizării resurselor, minimizarea efortului uman de programare și automatizarea operațiilor manuale în cât mai mare măsură, în toate fazele de pregătire și execuție a programelor" [3]. Materialul din acest capitol este preluat din [2], dreptul de preluare/multiplicare aparține autorului.

Sistemul de operare pune la dispoziția utilizatorilor (operatori, programatori etc.) și o interfață concretizată într-un *interpretor al comenzielor utilizatorului* exprimată cu ajutorul unui *limbaj de comandă*. Toate sistemele de operare moderne (UNIX, System, Windows* etc.) oferă și o interfață grafică, comenzi fiind selectate din meniuri ierarhice folosind dispozitive de interacționare grafică sau tastatura. Totuși, puterea limbajului de comandă nu poate fi atinsă numai cu ajutorul elementelor grafice.

Interfața dintre sistemul de operare și programele utilizatorului (numite și lucrări - eng. *jobs*) este asigurată de o colecție de "instructiuni extinse" ale sistemului de operare numite *apeluri sistem*. Folosind apele sistem, un program utilizator poate crea, utiliza și sterge diverse obiecte gestionate de sistemul de operare. Cele mai importante obiecte gestionate de un sistem de operare modern sunt *procesele și fișierele*.

Procesul (eng. *task*) reprezintă conceptul cheie al oricărui sistem de operare. *Un proces este o entitate dinamică care corespunde unui program în execuție*. Procesele sunt fie procese sistem, fie procese utilizator. Procesele sistem sunt asociate unor module ale sistemului de operare, iar procesele utilizator sunt asociate unor programe utilizator care pot să creeze alte procese utilizator sau să lanseze pentru executare procese sistem. Un proces (numit *proces tată*) poate crea unul sau mai multe procese (numite *procese fiu sau descendenți*). În sistemele multiutilizator fiecare proces este caracterizat de identificatorul proprietarului (utilizatorului).

Un fișier (eng. *file*) este un sir de caractere terminat printr-o marcă de sfârșit de fișier (EOF - eng. *End Of File*). Fișierul poate fi stocat pe disc, în memorie etc. Una din funcțiile importante ale unui sistem de operare este aceea de a "ascunde" dificultatea lucrului cu echipamentele periferice. Astfel, sistemul de operare oferă apele sistem pentru lucru cu fișiere: creare, ștergere, citire, scriere

etc. Fișierele pot fi grupate într-un catalog (eng. *directory, folder*) și pot fi caracterizate de anumite atribute (proprietar, dimensiune, data ultimului acces în scriere, coduri de protecție etc.).

Modulele software pentru tratarea cererilor de intrare/ieșire de către sistemul de operare se numesc *drive*. Fiecare dispozitiv periferic are asociat un driver. În general, orice driver menține o coadă a cererilor de intrare/ieșire lansate de unul sau mai multe procese și pe care le prelucrează într-o anumită ordine (în funcție de momentul lansării cererii sau conform unei liste a priorităților). Un driver este răspunzător de satisfacerea cererilor de transfer de informație, de tratarea erorilor ce pot apărea la realizarea unei operații fizice, și.a. Un program utilizator poate efectua operații de intrare/ieșire la nivelul unui driver, totuși programul său nu mai este independent de dispozitiv. De aceea, pentru operațiile de intrare-ieșire se utilizează apelurile sistem sau diferitele proceduri specializate puse la dispoziție de mediile de programare.

Primele sisteme de operare realizau prelucrarea pe loturi de programe (eng. *batch mode*). Utilizatorul nu comunica direct cu sistemul de calcul; acesta funcționa sub controlul unui operator uman specializat. Operatorul avea sarcina de a asigura resursele externe necesare unei lucrări (montarea benzilor magnetice, pornirea și oprirea diverselor echipamente periferice). De asemenea, operatorul asigura și introducerea lucrărilor în sistem. Comunicarea operațiilor de executat, se realiza prin intermediul unei interfețe de tip alfanumeric ce utiliza un limbaj de comandă pentru descrierea ordinelor adresate sistemului, precum și pentru specificarea acțiunilor necesare tratării erorilor.

Primele sisteme de acest tip funcționau în regim de *monoprogramare*, un singur program fiind încărcat în memorie la un moment dat. Caracteristica de bază a acestui mod de prelucrare o reprezintă imposibilitatea intervenției utilizatorului pentru a interacționa cu programul său.

Dintre concepțele implementate pentru creșterea performanțelor și mărirea eficienței utilizării resurselor sistemelor de calcul, un rol important l-a avut *multiprogramarea*. În sistemele cu multiprogramare, la un moment dat, în memorie se află încărcate, pentru executare, mai multe procese (programe în executare), ce concurează, pe baza unei scheme de priorități, pentru accesul la anumite resurse ale sistemului. Când o resursă este retrasă unui proces (la încheierea acestuia sau la apariția unui proces cu prioritate mai mare), aceasta poate fi imediat alocată unui proces solicitant.

Sistemele cu multiprogramare sunt din ce în ce mai complexe. Ele au de rezolvat probleme dificile privind: alocarea optimă a resurselor, evitarea interblocărilor, protecția utilizatorilor (între ei) și protecția sistemului (în raport cu utilizatorii).

În sistemele uniprocesor, execuția mai multor programe în regim de multiprogramare pare simultană din punctul de vedere al utilizatorului, dar la un moment dat, există doar un singur proces activ în sistem. Totuși, în sistemele multiprocesor sau multicalculator, două sau mai multe procese pot fi active simultan, ele fiind prelucrate de procesoare diferite.

Un alt mecanism important este *multiprocesarea*. Acesta constă în multiprogramarea a două sau mai multe procese având un obiectiv comun. Într-un *sistem de operare multiproces* (eng. *multitasking*) procesele pot comunica între ele și își pot sincroniza activitățile. Sistemele Microsoft bazate pe tehnologia NT, sistemele UNIX și Linux sunt sisteme de operare multiproces.

Mulțimea funcțiilor și modul de realizare a acestora definesc caracteristicile unui sistem de operare. Aceste caracteristici pot fi utilizate pentru a clasifica și compara sistemele de operare. Cele mai importante caracteristici sunt:

- i) modul de introducere a programelor în sistem;
- ii) modul de planificare a proceselor;
- iii) numărul de programe prezente simultan în memorie;
- iv) modul de utilizare a resurselor;
- v) numărul de utilizatori ce pot folosi sistemul în același timp.

După modul de introducere a lucrărilor (programelor) în sistem, se disting sisteme de operare seriale ce acceptă introducerea lucrărilor de la un singur dispozitiv de intrare, sisteme de operare paralele ce admit introducerea lucrărilor de la mai multe dispozitive de intrare precum și sisteme de operare cu introducerea lucrărilor de la distanță.

După modul de planificare a lucrărilor pentru execuție, există sisteme de operare orientate pe lucrări ce admit ca unitate de planificare lucrarea (eng. job), alcătuită din unul sau mai multe programe successive ale aceluiași utilizator, precum și sisteme de operare orientate pe proces care admit ca unitate de planificare procesul.

După numărul de programe prezente simultan în memoria principală se evidențiază sistemele cu monoprogramare și sistemele cu multiprogramare.

După numărul de utilizatori acceptă și simultan de sistemul de operare, există sisteme monoutilizator (calculatoare personale, stații de lucru, etc.) și sisteme multiutilizator.

După modul de utilizare a resurselor se disting:

- a) sisteme de operare cu resurse alocate, în care resursele necesare lucrărilor sunt afectate acestora pe toată durata executării;
- b) sisteme de operare în *temp real* care permit controlul executării lucrărilor într-un interval de timp specificat;
- c) sisteme cu resurse partajate (divizate), în care resursele necesare lucrărilor sunt afectate acestora periodic pe durata unor *cuante* de timp.

Când resursa partajată este timpul unității centrale, sistemul de operare devine cu *temp partajat* (eng. *time sharing*). În general, sistemele de operare din această clasă asigură utilizarea eficientă a resurselor sistemelor de calcul conversaționale în care accesul la resursele sistemului poate fi:

a) direct - pentru asigurarea unui control direct și permanent asupra unui set de terminale pe baza unui program utilizator; un caz particular al acestor sisteme îl reprezintă sistemele interactive în *temp real*, în cadrul cărora se cere o valoare maximă prestabilită a timpului de răspuns;

b) multiplu - pentru accesul simultan, al unui număr mare de utilizatori, la resursele hardware și software ale sistemului; acest tip de acces apare când sunt cel

puțin două terminale în sistem, iar fiecare utilizator lucrează cu programul său într-o regiune de memorie (partiție) diferită de regiunile celorlalți utilizatori, protejată printr-un mecanism software sau hardware;

c) în timp partajat (*time-sharing*) - în care alocarea timpului de acces se realizează pe baza unor cuante de timp fixe sau variabile, de ordinul milisecundelor, utilizatorii având impresia că lucrează simultan cu sistemul;

d) la distanță - în care prelucrarea se produce de către mai multe calculatoare asupra datelor care sunt distribuite în mai multe colecții de date dispuse geografic (eng. *distributed data bases*).

II.2. Structura sistemelor de operare

Un sistem de operare, în forma cea mai simplă, apare ca o colecție de proceduri cu funcții precise și cu o interfață bine precizată între acestea. Serviciile (apelurile sistem) furnizate de sistemul de operare, sunt solicitate prin încărcarea unui regisztr (ale UCC) cu informația necesară sau depunerea acestei informații în memoria stivă și apoi "provocarea" unei intreruperi cunoscută sub numele *apel supervisor* sau *apel nucleu*. Acest apel determină trecerea sistemului de calcul din *mod utilizator* în *mod nucleu* (supervizor) și transferă controlul sistemului de operare. Sistemul de operare analizează parametrii apelului pentru a-l identifica, iar apoi apelează procedura de serviciu necesară. Această descriere sugerează scheletul unui sistem de operare:

- i) un program ce invocă o procedură de serviciu;
- ii) o bibliotecă de proceduri de serviciu ce corespund apelurilor sistem ;
- iii) o bibliotecă de proceduri utilizare pentru procedurile de serviciu [3].

Pentru a-și îndeplini funcțiile, majoritatea sistemelor de operare sunt structurate pe două straturi:

1. *stratul logic* ce asigură interfața între utilizator și sistem (prin comenzi, instrucțiuni extinse și mesaje) și
2. *stratul fizic* reprezentat de rutinele de tratare a intreruperilor software. Ultimul strat este apropiat de hardware și, în general, transparent pentru utilizator.

Tendința în realizarea sistemelor de operare moderne este de a implementa cea mai mare parte a funcțiilor sistemului de operare sub formă de procese utilizator. Pentru a solicita un serviciu, un proces utilizator (numit și *proces client*) emite o cerere unui proces de serviciu, care rezolvă solicitarea și transmite clientului răspunsul. Această comunicație între clienți și procesele de serviciu este asigurată de nucleul sistemului de operare.

Deoarece calculatoarele personale sunt din ce în ce mai mult răspândite, iar cel mai popular și stabil sistem de operare pentru acest tip de sisteme de calcul este MS-DOS (*MicroSoft-Disk Operating System*), vom prezenta mai multe detalii privind acest sistem de operare. Mai întâi se prezintă componentele sistemului MSDOS, iar apoi etapele încărcării în memoria RAM a acestuia. Această tratare prezintă avantajul deprinderii lucrului cu sistemele de operare Windows (inclusiv Windows 2000) în mod comandă.

Calculatoarele personale compatibile MS-DOS au două sisteme de control. La nivel fizic se află monitorul ROM, denumit și ROM-BIOS. Sistemul de operare MS-DOS asigură al doilea nivel de control al unui calculator personal. Oricare alt sistem de operare, deși poate realiza punerea în lucru a resurselor fizice, în trepte (vezi sistemele MS-Windows), asigură tot al doilea nivel de control.

Monitorul ROM-BIOS este o colecție de subprograme stocate în memoria ROM, furnizate de firma producătoare ca o parte a sistemului hardware. BIOS (eng. *Basic Input/Output System*) este compus din:

- a) programul de autotestare;
- b) suportul software (drivere) pentru: consola monocrom, tastatură, imprimantă, dispozitive periferice auxiliare și ceasul sistemului;
- c) rutina de pornire la inițializarea sistemului;
- d) programul încărător al primului sector al discului sistem (dischetă sau disc rigid). Sistemul BIOS trebuie să asigure independența sistemului de operare de partea de hardware.

Sistemul de operare MS-DOS este mai performant decât monitorul ROM-BIOS și asigură o nouă interfață pentru programul de aplicație (prin intermediul apelurilor sistem MS-DOS). El are cinci componente principale: *încărătorul*, *extensia BIOS*, *nucleul MS-DOS*, *interfața cu utilizatorul*, *programele utilitare*. Componentele cheie sunt localizate în trei fișiere: extensia BIOS în fișierul *IO.SYS* (respectiv *IBMBIO.SYS* pentru varianta IBM), nucleul MS-DOS în fișierul *MSDOS.SYS* (respectiv, *IBMDOS.SYS*), interfața utilizator în fișierul *COMMAND.COM*. Pentru sistemul de operare Windows 2000, acest fișier este situat în catalogul *WINNT/System32*, iar pentru Windows 95 și Windows 98, în catalogul Windows al discului de pe care se încarcă sistemul de operare. *Command* este o aplicație MS-DOS.

Încărătorul este un program foarte scurt localizat în primul sector al discului sistem (discul ce conține sistemul de operare care va fi încărcat). El are rolul de a încărca sistemul de operare la pornirea calculatorului sau la repornirea acestuia folosind, repetat, combinația de taste: *CTRL+ALT+DEL*. Acest program încarcă următoarele două părți (*IO.SYS* și *MSDOS.SYS*) în memorie. Pentru sistemele de operare Windows, prima activare a acestei combinații duce la afișarea listei taskurilor active.

Fișierul *IO.SYS* conține driverele noi ce le extind sau le înlocuiesc pe cele din ROM-BIOS, rutina de inițializare a acestor drivere precum și un program încărător evoluat, pentru restul sistemului de operare, numit *SYSINIT*.

Nucleul MS-DOS, din fișierul *MSDOS.SYS*, este un intermediar între BIOS și programul de aplicație. Acesta oferă o interfață logică pentru programul de aplicație (API). Folosind API, un program de aplicație devine independent de sistemul hardware fără a pierde posibilitatea de acces la tastatură, monitor, imprimantă sau fișiere sistem.

Alături de interfața cu programul de aplicație, nucleul MS-DOS îndeplinește și următoarele funcții: gestiunea programelor, gestiunea memoriei și gestiunea fișierelor. Pentru programele de aplicație, MS-DOS utilizează alocarea memoriei fizice, sub formă de blocuri de memorie numite *paragrafe*. Gestiunea

executării programelor presupune încărcarea și inițierea execuției programului, precum și revenirea în sistemul de operare la terminarea programului.

Cu toate că MS-DOS nu este un sistem de operare multiproces, el poate asigura într-o oarecare măsură comunicarea între procese utilizând aşa numitele conducte (eng. *pipes*). Acestea (prezente și la sistemul de operare UNIX, de unde au fost împrumutate), sunt fișiere (temporare) de tip special ce conțin date scrise de un program pentru a fi citite de un alt program. De exemplu, comanda MS-DOS

`C:\>dir | more`

folosește o conductă MS-DOS pentru afișarea ecran cu ecran a cuprinsului catalogului curent.

O aplicație MSDOS poate avea acces la un fișier în două moduri: prin intermediul blocului de control al fișierelor și prin specificarea unui sir ASCII, conținând numele unității de disc, numele complet al catalogului din care (va) face parte fișierul, numele și extensia fișierului. Numele unității de disc este compus dintr-o literă de identificare și semnul ":". Un catalog (eng. *directory*) este un cuprins al discului. Există un catalog de bază, notat prin "\\" numit catalog rădăcină (eng. *root*⁷) de la care se ramifică toate celelalte subcataloage. Numele fișierului poate fi de la unul la opt caractere, scris atât cu litere mari cât și cu litere mici. Automat literele mici sunt convertite de MS-DOS în litere mari. Pentru cataloagele Windows (redenumite dosare [eng. *folder*]) se poate utiliza un număr mai mare de caractere.

Extensia este formată dintr-un punct și zero, unul, două sau trei caractere. Extensiile sunt optionale. Există câteva extensii cu semnificație specială (EXE - aplicații relocabile, COM - aplicații nerelocabile, BAT - fișiere ce conțin comenzi, SYS - fișiere sistem, TXT - fișiere ce conțin informație în reprezentare externă. BMP - fișiere imagine în format matriceal etc).

Sistemul de operare MS-DOS va aloca unitatea de disc, catalogul și fișierul creat sau deschis și îi va asocia un identificator logic de fișier. Orice acces relativ la fișier (citire, scriere, închidere etc.) poate fi realizat numai folosind acest identificator. Pentru un astfel de fișier, sistemul de operare MS-DOS construiește un bloc de parametri în zona de memorie proprie aplicației ce folosește fișierul. Numărul maxim de fișiere deschise simultan este specificat în comanda de configurare **FILES**.

Interfața utilizatorului cu sistemul MS-DOS (reprezentată de programul **COMMAND.COM**) este compusă din trei părți: *partea rezidentă*, *partea tranzitorie*, *componenta de initializare*. Partea rezidentă conține codul necesar prelucrării comenzi de oprire (**CTRL+C** și **CTRL+Break**), codul pentru erorile de intrerupere, precum și codul pentru încărcarea părții tranzitorii la nevoie. Partea tranzitorie preia comenziile de la tastatură sau din fișierele de comenzi indirekte (.BAT). Ea conține codul aferent comenziilor MS-DOS interne. Fiind stocată în zona memoriei convenționale, se poate întâmpla ca programul de aplicație, dacă este prea mare, să suprascrie partea tranzitorie a programului COMMAND.COM.

⁷ Denumirea *root* nu trebuie confundată cu utilizatorul *root* al sistemelor de operare Unix/Linux.

În acest caz, la terminarea programului partea rezidentă va reîncărca partea tranzitorie. Componenta de inițializare este încărcată numai la pornirea sistemului și conține codul pentru apelarea fișierului **AUTOEXEC.BAT**. După executarea comenziilor din acest fișier, componenta de inițializare este abandonată.

Comenzile acceptate de programul **COMMAND.COM** pot fi de tip intern, extern, directivă sau descriptive. *Comenzile interne* (de ex. **Chdir (Cd)**, **Cls**, **Copy**, **Date**, **Del (erase)**, **Dir**, **Echo**, **Exit**, **Loadhigh (lh)**, **Mkdir (Md)**, **Path**, **Rename**, **Rmdir (Rd)**, **Set**, **Type**, **Ver**) sunt comenzile executate de codul intern al programului **COMMAND.COM**. Ele sunt valabile și sub sistemele de operare Windows în modul comabdă (eng. *Ms-Dos prompt, Command prompt*). *Comenzile externe* (de ex. **Append**, **Command**, **Dblspace**, **Diskcopy**, **Doskey**, **Edit**, **Fdisk**, **Find**, **Format**, **Graphics**, **Help**, **Mem**, **More**, **Print**, **Qbasic**, **Sort**, **Tree**, **Undelete**, **Unformat**, **Xcopy**) sunt acele comenzi executate de un cod extern interfeței **COMMAND.COM**, având același tratament precum programele de aplicație (fișiere cu extensiile .EXE și .COM). De obicei aceste programe (utilitare) se află în catalogul "C:\DOS". Comenzile externe ale sistemului Windows 2000, inclusiv cele specifice, sunt situate în dosarul WINNT\System32. *Comenzile descriptive* (de ex. **Break**, **Buffers**, **Country**, **Device**, **Dos**, **Files**, **Lastdrive**) pot fi înregistrate în fișierul **CONFIG.SYS** pentru a defini anumiți parametri ai sistemului de operare. *Directivele* (de ex. **Call**, **For**, **Goto**, **If**, **Rem**, **Shift**) sunt folosite alături de comenzile interne și externe în *fișiere de comenzi indirecte* (eng. *batch files or script files*). Fișierele *batch* (.BAT) sunt fișiere text ce conțin o listă de comenzi interne, externe, directive sau specificații ale altor fișiere *batch*. Interpretorul din partea tranzitorie a programului **COMMAND.COM** citește fișierele *batch* linie cu linie, analizează propoziția curentă (descrișă prin verbul comenzi și, eventual, parametri) și execută operația asociată comenzi.

II.3. Utilizarea microcalculatoarelor personale folosind MS-Dos/Command Prompt

Cele mai importante comenzi pe care utilizatorul le poate folosi în conversația sa cu un sistem de operare, folosind interfața de tip text sunt de următoarele tipuri: a) comenzi pentru deschiderea sau închiderea unei sesiuni de lucru; b) comenzi pentru execuția unui program (lansarea unei comenzi); c) comenzi informaționale; d) comenzi pentru lucrul cu fișiere și cataloage; e) comenzi pentru lucrul cu volume; f) comenzi speciale (pentru configurare, administrare și întreținere sistem).

Comenzile MS-DOS sunt de două tipuri: interne și externe. Pentru a putea fi executată o comandă externă, pe discul sistem trebuie să existe un fișier executabil ce corespunde comenzi respective. Fiecare comandă MS-DOS transmite, în momentul terminării, un cod cu o valoare între 0 și 255. De regulă, un cod cu valoare nenulă indică un comportament nesatisfăcător. Această valoare este depusă în variabila globală **ERRORLEVEL**. Prin analiza acestei valori într-un program *batch* se poate lua o decizie în urma executării unei comenzi.

Mulțimea comenzilor pentru deschiderea/inchiderea unei sesiuni de lucru este vidă. Odată încărcat sistemul de operare MS-DOS, utilizatorul are acces necondiționat asupra tuturor resurselor sistemului de calcul. Pentru sistemul de operare Windows 2000, deschiderea sesiunii de lucru are loc după introducerea identificatorului (de acces) și a parolei, în caseta de dialog inițială sau cea declanșată prin acțiunea Start/ShutDown/Log off... Comenzile MS-DOS interne, disponibile și pentru Windows 2000, vor fi în continuare marcate prin "*".

Comenzi informaționale

În această categorie intră comenzi care după invocare afișează anumite informații privind starea sistemului. Uenele comenzi MS-DOS permit și modificarea anumitor parametri ai sistemului. Acestea vor fi descrise numai la grupa funcțională corespunzătoare. Se utilizează simbolurile: | - descrie o nouă variantă, [..] - entitate cu caracter opțional, *cale* - desemnează adresa relativă sau absolută a unui catalog.

Lista comenzilor informaționale cuprinde:

1. **HELP** [comanda]
2. **MEM** [/c|/d|/f|m *nume_program*][/page]
3. * **VER**
4. * **VOL**

Comenzi pentru lucrul cu fișiere și cataloage

Această categorie de comenzi permite efectuarea operațiilor globale asupra fișierelor (creare, ștergere, copiere, concatenare, afișare, tipărire, redenumire) și asupra cataloagelor [eng. *directories, folders*] (creare, ștergere, navigare în sistemul de fișiere). De asemenea există comenzi de informare asupra fișierelor și cataloagelor.

Principalele comenzi relative la fișiere sunt:

1. **ATTRIB** [+r | -r] [+a | -a] [+s | -s] [+h | -h] [[d:][*cale*]*nume_fișier*] [/s]
2. * **DIR** [d:][*cale*][*nume_fișier*] opțiuni
opțiuni:=[/p] [/w] [/a [[:]attribute]][/o[[:]ordine_sortare]] [/s] [/b] [/l] [/c[b]]
3. * **COPY** [/y|-y] [/a|/b] *sursa* [/a|/b] [+ *sursa* [/a|/b] [+...]] [destinație [/a|/b]] [/v]
4. * **TYPE** [d:][*cale*] *nume_fișier*
5. **PRINT** [/d: *port*] [/b: *lungime*] [/u: *temp*] [/m: *max_temp*] [/s: *cuanta*] [/q: *dimc*] [/t] [[d:][*cale*]*nume_fișier*[...]] [/c] [/p]
6. * **RENAME** [d:][*cale*] *f1 f2* sau **REN** [d:][*cale*] *f1 f2*
7. * **DEL** [d:][*cale*] *nume_fișier* sau **ERASE** [d:][*cale*] *nume_fișier*
8. **UNDELETE** [[<BI>d:][*cale*] *nume_fișier*][/dt]/ds/dos]
UNDELETE [/list|/all|/purge [d:]/status|/load|/unload|/s[d]|td[-maxpoz]]
9. **COMP** [d:][*cale*][*nume_fișier*] [d:][*cale*][*nume_fișier*]

10. **FC** [/**a**] [/**c**] [/**I**] [/**lb***n*] [/**n**] [/**t**] [/**w**] [/*nnnn*] [d1:][*cale1*]*fisier1*
 [d2:][*cale2*]*fisier2*
FC /b [d1:][*cale1*]*fisier1* [d2:][*cale2*]*fisier2* (pentru copiere binară)
 Cele mai importante comenzi relative la cataloge sunt:
- 11. * **MKDIR** [d:]*cale* sau **MD** [d:]*cale*
- 12. * **CHDIR** [d:]*cale* sau **CD** [d:]*cale* , **CD ..** , **CD .** sau variantele
CHDIR.
- 13. * **RMDIR** [d:]*cale* sau **RD** [d:]*cale*
- 14. **TREE** [d:][*cale*][/f][/a]
- 15. **DELTREE** [/y] [d:]*cale*
- 16. **MOVE** [/y | -y] [d:][*cale*][[*fisier* [, [d:][*cale*]*fisier*[...]]] *destinație*
- 17. **XCOPY** [/y| -y] [d1:][*cale1*][*fisier1*] [d2:][*cale2*][*fisier2*]
 [/a|m] [/d: *ll-zz-aa*] [/p] [/s [/e]] [/v] [/w]
- 18. **JOIN** sau **JOIN** d1: d2:*cale* sau **JOIN** d1:/d
- 19. **SUBST** [d1:[d2:][*cale*]] sau **SUBST** d1: /d

Comenzi relative la volume

În cadrul sistemului de operare MS-DOS, comenziile de lucru relative la volume se reduc la comenziile de utilizare a discului rigid și a dischetelor. Sintaxa principalelor comenzi relative la discuri este prezentată sintetic mai jos.

1. **LABEL** [d:][*eticheta*]
2. **DISKCOMP** [d1: [d2:]] [/1] [/8]
3. **DISKCOPY** [d1: [d2:]] [/1] [/v] [/m]
4. **FORMAT** d:[/v [:*eticheta*]][/q] [/u] [/f: *dimensiune*][/b] [/s] sau
 FORMAT d:[/v [:*eticheta*]][/q] [/u] [/t: *piste* /n: *sectoare*] [/b] [/s] sau
 FORMAT d:[/v [:*eticheta*]][/q] [/u] [/1] [/4] [/b] [/s] sau
 FORMAT d:[/q] [/u] [/1] [/4] [/8] [/b] [/s]
5. **UNFORMAT** d:[/l] [/test][/p]

Comenzi speciale

Această secțiune descrie o parte din comenziile MS-DOS care trebuie executate cu foarte mare atenție. Comenziile speciale sunt împărțite în următoarele categorii:

- *comenzi de configurare dinamică* (**PATH**, **APPEND**, **DATE**, **TIME**, **PROMPT**, **CLS**);
- *comenzi de configurare a sistemului de operare* (Realizarea unui fișier **CONFIG.SYS**);
- *comenzi de administrare a sistemului de calcul.*

Sintaxa principalelor comenzi de configurare dinamică este prezentată mai jos. Acestea sunt destinate specificării căilor de căutare a fișierelor, stabilirii promptului, specificării datei și a orei, etc.

1. * **PATH** [[d:]*cale*[...]]
2. **APPEND** [[d:]*cale*[...]]
3. * **DATE** [*ll-zz-aa*]
4. * **TIME** [*hh:mm:ss:xx*]] [a | p]

5. * **PROMPT** [*specificator_prompt*]
6. * **CLS**
- Sintaxa principalelor **comenzi pentru crearea unui fișier CONFIG.SYS** este prezentată în continuare. Sunt disponibile și comenzi suplimentare de configurare folosind tehnica meniurilor. Acestea nu sunt prezentate aici.
7. **BREAK** = [on | off]
8. **BUFFERS** = *n[, m]*
9. **COUNTRY** = *xxx[,yyy][,d:][cale]nume_fișier*]
10. **DOS** = **HIGH** | **LOW** [**UMB** |, **NOUMB**] sau
DOS = [**HIGH**, | **LOW**,] **UMB** | **NOUMB**
11. **DEVICE** = [*d:*][*cale*]*nume_fișier* [*parametri*]
12. **DEVICEHIGH** = [*d:*][*cale*]*nume_fișier* [*parametri*]
DEVICEHIGH [[/I:*reg1[,min1][; reg2 [, min2]...]*] [/S]] =
[d:][*cale*]*nume_fișier* [*parametri*]
13. **FCBS** = *n*
14. **FILES** = *n*
15. **INSTALL** = [*d:*][*cale*]*nume_fișier* [*parametri*]
16. **LASTDRIVE** = **L**
17. **STACKS** = *n, dim*
18. **SHELL** = [[*d:*]*cale*]*nume_fișier* [*parametri*]

Pentru a inițializa sau modifica valoarea unei variabile globale se utilizează următoarea sintaxă:

SET *identifier* = [*sir*]

iar pentru afișarea variabilelor globale existente se folosește comanda **SET** fără argumente.

Parametrul *identifier* precizează variabila care va fi inițializată sau a cărei valoare va fi modificată. Toate literele mici sunt convertite la litere mari. Parametrul *sir* descrie șirul de caractere care va fi asociat variabilei. Dacă acest șir este vid, atunci variabila va fi ștearsă din zona de memorie rezervată.

Observație: Comanda **SET** poate fi folosită și în programele *batch* în legătură cu parametrii poziționali. Acest aspect va fi ilustrat mai jos.

Comenzile de administrare a sistemului de calcul sunt comenzi MS-DOS care acționează asupra volumelor de disc în vederea partiționării, verificării stării unui volum, refacerii unui volum, configurării echipamentelor I/E de tip caracter: tastatură, imprimantă etc. Aceste comenzi se utilizează numai la nevoie și cu foarte mare atenție. Utilizatorul trebuie să cunoască foarte bine contextul în care se aplică comenzile precum și efectul lor. Spre exemplificare, prezentăm câteva comenzi din această categorie.

19. **FDISK** sau **FDISK /status**
20. **SCANDISK** [*parametri*]
21. **CHKDSK** [*d:*][[*cale*]*nume_fișier*] [/F] [/V]
22. **INTERLNK**
23. **INTERSVR**
24. **MODE**
25. * **CTTY** *dispozitiv*

Comenzile sistemului de operare pot fi introduse de utilizator (ca mai sus) sau pot fi grupate în programe (fișiere de comenzi indirekte) numite programe *batch* executabile de către un procesor de comenzi. Un fișier de comenzi indirekte poate fi creat cu orice editor de texte și, de obicei, numele său are o extensie predefinită (**BAT**).

Programele *batch* utilizează noțiuni caracteristice limbajelor de nivel înalt cum ar fi: procedură, variabile, parametri, structuri de control și operații caracteristice macrogeneratoarelor.

Un fișier de comenzi indirekte MS-DOS este un fișier text ce conține mai multe tipuri de construcții: comenzi MS-DOS propriu-zise, etichete, caractere speciale (| < > % @), parametri poziționali, parametri simbolici, variabile locale și directive. Orice etichetă este un sir de maximum 9 caractere (litere sau cifre), primul caracter fiind ":". O etichetă apare (în momentul definirii) pe o singură linie, la începutul rândului.

Caracterul special | între două comenzi A și B specifică legarea în conductă (eng. *pipe*) a celor două comenzi. Caracterele < și > specificate într-o comandă semnifică comutarea intrării, respectiv a ieșirii standard a comenzi. Caracterul @, prezent ca prim caracter într-o linie, indică prelucrarea liniei respective de către procesorul de comenzi, dar linia nu este afișată pe ecran.

Parametrii formali au forma %0, %1, ..., %9, și permit specificarea unor argumente în linia de apel a programului *batch*. Aceștia se mai numesc și parametri poziționali deoarece sunt referiți prin ordinea în care apar în linia de comandă.

O variabilă locală are forma %%c și este utilizată ca variabilă de ciclare într-o directivă **FOR**. Identificatorul c este format dintr-un singur caracter.

Parametrii simbolici sunt acei parametri care primesc valori prin comanda MS-DOS **SET** utilizată înainte de apelul programului *batch*. Pentru a avea acces la valorile variabilelor globale (stabilite prin comenzi **SET** în afara programului *batch*), numele acestor variabile trebuie delimitat de două caractere %. Prin intermediul acestui tip de parametri, două sau mai multe programe *batch* pot comunica între ele.

Programele *batch* pot fi compuse din secvențe liniare, secvențe repetitive și apeluri ale altor programe *batch*. Pentru realizarea structurilor alternative, procesorul de comenzi MS-DOS recunoaște mai multe comenzi numite *directive*. Regulile sintactice ale celor mai importante directive sunt:

1. **CALL** [d:][cale] *program_batch* [parametri]
2. **GOTO** [:]*id*
3. **IF** [**NOT**] **ERRORLEVEL** *număr comandă*
 IF [**NOT**] *text1 == text2 comandă*
 IF [**NOT**] **EXIST** *fișier comandă*
4. **PAUSE** [*mesaj*]
5. **FOR** %%c **IN** *mulțime DO comandă*
6. **SHIFT**
7. **ECHO** [**ON** | **OFF** | *mesaj*]
8. **REM** [*comentariu*]
9. **CHOICE** [/c[:]șir_de_taste] [/n] [/s] [t[:]x, nn] [*text*]

II.4. Utilizarea microcalculatoarelor personale folosind sistemul de operare UNIX/Linux

Sistemul de operare UNIX a fost elaborat în anul 1969 (autori: Ken Thompson și Dennis Ritchie - *Laboratoarele Bell* din New Jersey, SUA) și până în prezent a cunoscut mai multe extensii. Pentru microcalculatoare personale, cea mai cunoscută implementare este cunoscută sub numele de Linux.

Cele mai importante caracteristici ale sistemului de operare UNIX sunt:

- este un sistem de operare cu divizarea timpului, multiproces și multiutilizator;
- asigură protecția fișierelor și a modului de executare a programelor prin existența unor parole și drepturi (coduri) de acces;
- promovează modularitatea aplicațiilor;
- operațiile de intrare/ieșire sunt integrate în sistemul de fișiere, prin realizarea așa-numitelor intrări/ieșiri generalizate;
- unitatea de planificare este procesul;
- gestiunea memoriei se face printr-un mecanism care permite schimbul de pagini (migrația sau eng. *swapping*) între memoria RAM și cea externă, gestionându-se spațiul afectat executării proceselor și controlându-se timpul de acces al proceselor în aşteptare;
- prin intermediul componentei SHELL, asigură o interfață simplă și interactivă. Componenta SHELL nu este integrată în nucleul sistemului de operare;
- este un sistem de operare portabil (fiind scris în limbajul C) activ pe foarte multe tipuri de sisteme de calcul, de la calculatoare personale până la calculatoare MIMD.

Sistemul de operare UNIX este alcătuit din trei părți majore: nucleul, sistemul de fișiere (ce cuprinde programele utilitare, programele aplicative și programele de gestiune a intrărilor și ieșirilor) și interfața cu utilizatorul (componenta SHELL). Relațiile între cele trei componente ale sistemului se realizează prin:

- apele sisteme;
- programe utilitare;
- funcții standard folosite de limbajul C;
- subprograme de gestiune a intrărilor/ieșirilor, furnizate odată cu sistemul și diferite de la un sistem de calcul la altul.

Sistemul de operare UNIX oferă utilizatorului interfețe organizate pe trei nivele: nivelul exterior nucleului (prin programe utilitare); nivelul intermediar oferit de funcțiile din biblioteca standard C și nivelul interior oferit de apele sistem.

Sistemul de operare UNIX fiind un sistem multiutilizator și cu divizarea timpului, impune existența unui utilizator special numit *administrator de sistem*, care ține evidența utilizatorilor, stabilește parolele și drepturile de acces și creează cataloagele asociate utilizatorilor.

Pentru fiecare utilizator, administratorul de sistem creează câte un catalog propriu, care poate conține atât fișiere ordinare (programe sau date), cât și subcataloage.

Numele unui fișier poate avea până la 14 caractere, cu excepția caracterului blanc. Sistemul UNIX face deosebire între litere mari și litere mici. Toate fișierele sunt structurate în cataloge, organizate arborescent, în vârful ierarhiei (la rădăcina arborelui) aflându-se catalogul rădăcină (eng. *root*) notat prin '/'.

Specificarea numelui fișierului se poate face în două moduri: **absolut** - pornind de la rădăcină sau **relativ** - pornind de la poziția curentă. Numele complet al fișierului conține și calea de acces (catalogul din care face parte acesta).

Deschiderea unei sesiuni de lucru se realizează folosind comanda **login**. Sistemul va solicita numele utilizatorului (eng. *username*) precum și parola sa (eng. *password*). Un utilizator poate să-și schimbe parola folosind comanda **passwd**. Pentru terminarea sesiunii de lucru se utilizează combinația de taste **CTRL+d**. Pentru documentare se poate utiliza comanda **man** care prezintă informații despre anumite entități ale sistemului de operare. De exemplu, comanda **\$man man**, descrie structura manualelor UNIX precum și modul lor de consultare.

După deschiderea sesiunii de lucru, utilizatorul are acces la două grupe de fișiere: fișierele create de el însuși și fișierele furnizate de sistem drept comenzi sistem.

Executarea unei comenzi

În general, o comandă UNIX poate fi privită ca o succesiune de zone separate prin spații, de forma

comandă opțiuni expresii fișiere

unde: "comandă" este numele propriu-zis al comenzi, "opțiuni" reprezintă o secvență de opțiuni (o opțiune UNIX este reprezentată printr-o literă precedată sau nu de semnele "+" sau "-"), "expresii" reprezintă unul sau mai multe siruri de caractere cerute ca argumente pentru comanda respectivă, iar "fișiere" specifică unul sau mai multe fișiere.

Delimitarea anumitor argumente se poate face prin apostrofuri și ghilimele. În acest sens, trebuie să se respecte următoarele patru reguli de delimitare:

1. dacă în sirul de caractere nu apare nici unul din caracterele ' sau ", atunci se poate delimita fie prin ', fie prin ";
2. dacă în sirul de caractere apare apostroful, dar nu apare caracterul " , atunci delimitarea se poate realiza folosind caracterul ";
3. dacă în sirul de caractere apare caracterul ", dar nu apare apostroful, atunci delimitarea se realizează folosind apostrofuri;
4. dacă în sirul de caractere apar ambele caractere (' și ") atunci se va utiliza caracterul de evitare "\". Pentru prezența caracterului "\\" în sir, acesta se dublează.

Interpretorul de comenzi UNIX acceptă cel puțin o comandă pe linie. Dacă se dorește specificarea mai multor comenzi pe același rând, acestea trebuie separate prin caracterul ";".

Comenzi pentru lucrul cu procese

Spre deosebire de sistemul de operare MS-DOS, sistemul UNIX permite lansarea spre executare a unei comenzi (program) la un anumit moment de timp. Comanda UNIX ce face posibil acest lucru este disponibilă numai administratorului de sistem. Numele comenzi este **at** și primește ca argumente

timpul, data, factorul de repetare a lansării, precum și numele unui fișier care conține comanda sau comenzi ce se vor executa. De asemenea, folosind comenzi **nice**, **kill** și **sleep**, utilizatorul poate interveni asupra proceselor din sistem pentru a le modifica starea.

Comanda UNIX ce permite afișarea stărilor unor procese este **ps**. Cele mai uzuale opțiuni ale acestei comenzi sunt:

- e : listează toate procesele;
- f : produce o listare completă;
- l : produce o listare în format lung;
- p lista : afișează date doar despre procesele specificate în listă;
- t lista : afișează date doar despre procesele terminalelor din listă;
- u listă : afișează date doar despre procesele utilizatorilor din listă.

Informațiile despre fiecare proces, în formatul lung, sunt prezentate pe o linie sub forma unui tabel cu următoarele coloane:

F : *tipul procesului* (00 - proces terminat, 01 - proces sistem, 04 - proces suspendat de părantele său, 10 - proces încărcat în memorie, dar blocat);

S : *starea procesului* (R - proces în coada de așteptare (INTRERUPT), S - proces inactiv (**sleep** - mai puțin de 20 de secunde), I - proces inactiv (**idle** - peste 20 de secunde), T - proces terminat, D - proces evacuat temporar pe disc, O - proces ACTIV).

UID : *proprietarul* procesului;

PID : *identificatorul* procesului;

PPID : *identificatorul procesului părinte* al acestui proces;

PRI : *prioritatea* procesului (valoare mică înseamnă prioritate mare);

TTY : *terminalul* de la care a fost lansat procesul (chiar dacă este lansat de la distanță, în urma unei sesiuni de tip **telnet** sau **ssh**);

TIME : *timpul total* cât a fost ACTIV;

NICE : dacă *prioritatea* a fost stabilită folosind comanda **nice**;

ADDR : *adresa* din memorie la care se află încărcat procesul;

SZ : *dimensiunea* procesului;

STIME : momentul de *start* al procesului;

CMD : *comanda* care a lansat procesul.

Comanda **kill** emite un semnal de tip intrerupere către un proces. De obicei, acest semnal solicită terminarea procesului. Comanda are ca argumente un număr de semnal (precedat de -) și identificatorul procesului căruia i se adresează semnalul. Semnalul este un număr între 1 și 31 prin care este codificat tipul semnalului de intrerupere. Cele mai importante semnale sunt:

-2 : intrerupere la apăsarea tastei DEL;

-9 : oprire necondiționată;

-15 : semnal software de oprire;

-16-31 : semnale definite de utilizator.

În absența specificării unui semnal (ci numai a identificatorului procesului), implicit este considerat semnalul 15.

O altă facilitate a sistemului de operare UNIX privind execuția comenziilor o constituie lansarea în fundal (eng. *background*) a proceselor care nu sunt

conversaționale. Pentru a lansa un astfel de proces, linia de comandă trebuie să se încheie cu simbolul "&".

Comenzi informaționale

O mare parte din comenziile sistemului UNIX afișează și modifică anumite entități. Unele comenzi sunt numai informaționale. Printre acestea se pot enumera:

1. **ps** : afișează starea proceselor (descrișă mai sus);
2. **who** : afișează utilizatorii din sistem;
3. **logname** : afișează utilizatorul curent;
4. **whodo** : afișează informațiile despre utilizatori și procese;
5. **pwd** : afișează numele catalogului curent;
6. **ls** : afișează conținutul unui catalog;
7. **cal** : tipărește calendarul pentru anul specificat;
8. **du** : afișează numărul de blocuri conținute în fiecare fișier și catalog specificat ca argument;
9. **ipcs** : tipărește informații despre structurile active de comunicare evoluată între procese (cozi de mesaje, semafoare, zone de memorie partajată);
10. **df** : afișează numărul de blocuri disc libere și numărul I-nodurilor libere pentru un anumit sistem de fișiere, sau pentru toate sistemele de fișiere montate;
11. **file** : afișează tipul unui fișier specificat (text, executabil, catalog).

Comenzi relative la fișiere și cataloage

Comenziile UNIX ce acționează asupra fișierelor și cataloagelor permit navigarea în sistemul de fișiere (explorarea), copierea, ștergerea, fixarea atributelor, afișarea și tipărirea fișierelor precum și o serie de operații de căutare, comparare etc. Cele mai importante comenzi din această categorie sunt:

1. **cat** : afișează un fișier la ieșirea standard. Dacă sunt specificate mai multe fișiere, acestea vor fi afișate unul după altul.
2. **pg** : afișează paginat un fișier. Afișarea este întreruptă la apăsarea tastei "q".
3. **more** : afișează ecran cu ecran fișierele specificate (vezi și **pg**);
4. **cd** : schimbă catalogul curent;
5. **mkdir** : creează un nou catalog;
6. **rmdir** : șterge cataloagele specificate. Acestea trebuie să fie vide.
7. **rm** : șterge unul sau mai multe fișiere. Folosind opțiunea **-r** se pot șterge și cataloage ce nu sunt vide.
8. **mv** : mută sau redenumește fișierele și cataloagele. Comanda **mv** acționează conform următoarelor reguli:
 - i) Dacă sursa este un fișier existent, iar destinația este nume de fișier, atunci are loc operația de redenumire.
 - ii) Dacă sursa este un fișier, iar destinația este tot un fișier (existent), atunci fișierul existent este înlocuit cu fișierul sursă.
 - iii) Dacă sursa este un catalog, iar destinația este un nume, atunci catalogul este redenumit.

- iv) Dacă sursa este un catalog, iar destinația este un catalog existent, atunci mută catalogul sursă astfel încât să devină un subcatalog al catalogului existent.
 - v) Dacă sunt specificate unul sau mai multe fișiere în sursă, iar destinația este un catalog existent, atunci fișierele sunt mutate în acest catalog.
9. **cp** : copiază unul sau mai multe fișiere sursă într-un fișier destinație. Dacă destinația este numele unui catalog, atunci fișierele sunt copiate în catalogul specificat.
10. **mount/umount** : creează o intrare în tabela dispozitivelor (operația de montare) sau șterge intrarea din această tabelă (demontare). Montarea unui dispozitiv într-un catalog nevid, face inaccesibil (pe durata montării) conținutul acestuia. A se vedea și efectul comenzi **JOIN** din MS-DOS.
11. **chmod** : schimbă modul de acces al unuia sau mai multor fișiere. Numai proprietarul unui fișier sau un utilizator privilegiat poate schimba modul de acces.
12. **chown** : schimbă proprietarul unuia sau mai multor fișiere. Numai proprietarul curent sau un utilizator privilegiat poate modifica proprietarul unui fișier.
13. **cmp** : compară două fișiere;
14. **diff** : compară două fișiere și arată modificările care trebuie efectuate pentru a avea aceeași formă;
15. **diffdir** : afișează diferențele dintre două cataloage;
16. **find** : caută unul sau mai multe fișiere care satisfac anumite criterii

Comenzi relative la volume

1. **diskformat** : inițializează un disc (disc rigid sau dischetă) și îl formatează conform specificațiilor precizate în linia de comandă;
2. **badtrk** : cercetează suprafața discului pentru depistarea pistelor defecte;
3. **divvy** : divizează o partiziune a discului în mai multe diviziuni;

Alte comenzi

1. **clear** : șterge ecranul (fereastra terminal);
2. **date** : tipărește și modifică data curentă;
3. **stty** : fixează/afișează anumiți parametri pentru terminalul curent;
4. **fsck** : verifică și repară sistemul de fișiere (catalog /etc).

II.5. Utilizarea microcalculatoarelor personale folosind sisteme de operare de tip Windows

În ultimii ani, firma Microsoft a dezvoltat pentru calculatoare personale interfețe grafice similare sistemului de operare System - MacOS (sisteme de calcul Apple-Macintosh) respectiv interfeței grafice X-Window System a sistemelor de operare UNIX. Spre deosebire de interfețele: Windows 3.1 și Windows for Workgroups care asigură funcții superioare sistemului de operare MSDOS, dar se execută sub sistemul MSDOS, interfețele Windows dezvoltate după 1995, pot fi

considerate sisteme de operare pentru calculatoare personale care oferă însă și posibilitatea conectării în rețea a acestor calculatoare. WINDOWS NT/2000 este un sistem de operare adecvat atât serverelor cât și stațiilor de lucru dintr-o rețea.

Următoarele noțiuni sunt importante pentru operarea unui software cu interfață grafică:

- *Pictogramă* (eng. *icon*) - un simbol utilizat pentru a reprezenta grafic scopul și/sau funcția unei aplicații, catalog sau fișier.
- *Fereastră* (eng. *window*) - zonă dreptunghiulară distinctă pe ecran, delimitată de un chenar. O fereastră reprezintă un obiect deschis și este folosită pentru a afișa informații (text, grafică, meniuri etc.). Pot fi deschise mai multe ferestre simultan. O fereastră poate fi închisă (coresponde terminării aplicației care a deschis fereastra), redimensionată sau mutată pe suprafața de lucru.
- *Suprafață de lucru* (eng. *desktop*) - întregul ecran reprezentând zona de lucru. Pictogramele, ferestrele și bara de operații sunt afișate pe suprafața de lucru. A se observa că există și un subcatalog DESKTOP al catalogului WINDOWS.
- *Bara de meniuri* (eng. *menu bar*) – bara orizontală conținând numele meniurilor disponibile.
- *Bara de operații* – bara situată la baza suprafeței de lucru prestată de mediile de tip Windows. Conține butonul **Start** precum și butoane pentru toate programele și documentele deschise.
- *Bara cu instrumente* (eng. *toolbar*) – bara situată sub bara de meniuri a programelor din Windows, care afișează un set de butoane pentru executarea celor mai uzuale comenzi de meniu. Barele cu instrumente pot fi mutate sau ancorate pe orice latură a ferestrei program.
- *Dosar* (eng. *folder*) – container în care sunt stocate pe disc documente, fișiere program și alte dosare. Sinonim pentru catalog (eng. *directory*).
- *Subdosar* – Dosar inclus într-un alt dosar; sinonim pentru subcatalog. Toate dosarele sunt subdosare ale dosarului rădăcină.
- *Scurtătură* (eng. *shortcut*) - legătură rapidă către un obiect (fișier, disc etc.). Are asociată o pictogramă specială (o mică săgeată în colțul din stânga-jos).
- *Calculatorul meu* (eng. *My Computer*) - program de navigare prin resursele locale ale sistemului de calcul (discuri, imprimante, programe de configurare și de programare a lansării proceselor).
- *Coșul de gunoi* (eng. *Recycle Bin*) - un dosar special care stochează temporar obiectele șterse de utilizator. Dacă obiectele nu au fost șterse permanent atunci se pot refa (eng. *restore*).
- *Rețeaua locală* (eng. *Network neighborhood* sau *My Network places*) - program ce permite explorarea nodurilor rețelei locale (LAN) din care face parte sistemul.

- *Memoria Clipboard* - Folosind o parte din memoria RAM, obiectele (fișiere, dosare, text, desene etc.) pot fi mutate dintr-o structură în alta, pot fi multiplificate în cadrul sistemului sau pot fi înlăturate prin suprascrarea memoriei RAM. Mecanismul este disponibil în oricare program pentru care meniul de editare oferă funcțiile **cut** (mută conținutul în memoria clipboard), **copy** (copiază conținutul în memoria clipboard) și **paste** (copiază conținutul memoriei clipboard în locul specificat).

Lansarea spre executare a programelor este posibilă prin intermediul comenzi **RUN** din meniul **START**, prin activarea aplicației înregistrate în oricare submeniu al meniului **START** sau prin activarea pictogramei corespunzătoare.

Sistemul de operare Windows 2000 dispune de o colecție bogată de comenzi executabile în mod text. O mare parte a comenziilor MS-DOS prezentate anterior sunt utilizabile. În plus sunt prezente comenzi precum cele din tabelul următor.

ASSOC	afișează / modifică asocierea extensiilor fișierelor
AT	planifică comenzi și programe pentru a fi executate mai târziu
CACLS	afișează / modifică listele de control privind accesul la fișiere
CHKNTFS	afișează / modifică verificarea discului în momentul încărcării sistemului de operare
CMD	lansează o nouă instanță a interpretorului de comenzi
COLOR	stabilăște culorile implicate ale consolei
COMPACT	afișează / modifică compresia fișierelor din partiiile NTFS
CONVERT	convertește volume FAT în NTFS. Nu se poate converti discul curent.
SETLOCAL	inițiază localizarea variabilelor de mediu într-un fișier batch
ENDLOCAL	încheie localizarea variabilelor de mediu dintr-un fișier batch
FIND	caută un sir de caractere într-un fișier sau o listă de fișiere
FINDSTR	caută siruri în fișiere
FTYPE	afișează / modifică tipurile de fișiere utilizate în lista de asociere.
GRAFTABL	permite sistemului de operare să afișeze setul de caractere extins în mod grafic.
PUSHD	salvează dosarul curent și apoi îl modifică
POPD	reface valoarea anterioară acțiunii PUSHD
RECOVER	recuperează informația text de pe discuri cu defecte
REPLACE	înlocuiește fișiere
START	deschide o nouă fereastră pentru executarea unui program sau a unei comenzi
TITLE	stabilăște titlul ferestrei unei sesiuni cmd.exe
VERIFY	solicită verificarea scrierii corecte a datelor pe disc

Explorarea informațiilor stocate pe discurile unui sistem de calcul sub sistemul de operare Windows 2000 se poate realiza în mai multe moduri. „Aplicația” *MyComputer* permite accesul la resursele sistemului de calcul. Se va selecta discul dorit, iar apoi se va merge în adâncime, dar în fereastra aplicației sunt afișate fișierele din dosarul curent precum și sub-dosarele acestuia. Imaginea de ansamblu asupra întregii colecții de date se poate obține folosind programul *Windows Explorer* din subgrupul *Accessories* al grupului *Programs*.

Operațiile cu fișiere și dosare sunt naturale. Se pot executa fie prin intermediul meniurilor, cu mausul prin operația de tragere, fie prin intermediul memoriei *Clipboard* (pentru mutarea / copierea fișierelor și dosarelor).

II.6. Resurse logice privind programarea calculatoarelor

Pentru activitatea de programare sunt utile generatoarele de programe. Acestea transformă programul sursă într-un nou format [3]. Din această categorie fac parte: macrogeneratorul, asamblorul (relocabil, absolut), compilatoarele, interprotoarele, editorul de legături, bibliotecarul, editoarele de texte etc.

Macrogeneratorul analizează un text sursă conținând descrieri într-un limbaj special și prezintă la ieșire un fișier cu text scris în limbaj de asamblare, care poate fi prelucrat ulterior de asamblorul relocabil sau cel absolut. De exemplu, folosind TASM (al firmei Borland) se pot asambla programe în format (cod) Intel. Pentru a putea avea portabilitate între sistemele de tip Microsoft și Linux, pentru procesoare Intel și compatibile se poate utiliza NASM.

Programul sursă, scris de utilizator în limbaj de macroasamblare, este constituit dintr-un număr de linii. Fiecare linie conține o instrucțiune a limbajului de asamblare sau o directivă de macrogenerare. Rolul macrogeneratorului este de a expanda macroinstrucțiunile existente și a rezolva, pe cât posibil, blocurile condiționale.

Macrogeneratorul recunoaște și analizează: directivele de control numeric (tipul bazei de numerație), directivele de terminare (.END), directivele de asamblare condițională (.IF, .IFF, .IFT etc.), directivele de definire a macroinstrucțiunilor (.MACRO, .ENDM), directivele de control (mesaje de eroare), directivele de generare și substituire, directivele de apelare a macrodefinițiilor dintr-o bibliotecă etc.

Asamblorul relocabil transformă modulele sursă scrise în limbaj de asamblare într-un modul obiect relocabil, o tabelă de simboluri și un listing de asamblare. Asamblorul relocabil acceptă la intrare unul sau mai multe fișiere sursă în limbaj de asamblare obținute folosind macrogeneratorul sau scrise de utilizator. Ieșirile asamblorului constau dintr-un fișier obiect (.obj, .o etc.), un fișier de listare (.lst) și o tabelă de simboluri (.map).

Asamblorul absolut transformă modulele scrise în limbaj de asamblare (ieșire a macrogeneratorului sau scrise de utilizator) într-un program executabil (.tsk, .cmd, .out, etc.)

Compilatoarele efectuează translatarea programului sursă în program obiect relocabil. Pentru ca un astfel de modul să devină program executabil este necesară editarea legăturilor. Funcția principală a editorului de legături este de a transforma și înlănuiri modulele obiect relocabile rezultate în procesul de asamblare și/sau compilare pentru a obține un program executabil acceptabil de programul încărcător al sistemului de operare. În faza de editare a legăturilor pot fi incorporate și module obiect situate în biblioteci relocabile.

Un compilator este structurat în patru componente principale: analizor lexical, analizor sintactic și semantic, generator de cod și optimizator. Toate aceste componente gestionează un tabel de simboluri.

Analizorul lexical realizează o primă translatare a textului programului sursă într-un sir de entități lexicale ce constituie reprezentări interne ale unor categorii sintactice precum: cuvinte cheie, identificatori, constante, delimitatori, operatori etc. Astfel, în fazele următoare se poate lucra cu simboluri de lungime fixă și cu un număr mai mic de categorii sintactice. Analizorul lexical va completa tabelul de simboluri.

Analizorul sintactic și semantic verifică corectitudinea sintactică a instrucțiunilor și colectează atributele semantice ale categoriilor sintactice din program. Ieșirea analizorului sintactic și semantic o constituie o reprezentare codificată a structurii sintactice și un set de tabele ce conțin atributele semantice ale diferitelor categorii sintactice (simboluri, constante etc.)

Generatorul de cod va traduce ieșirea analizorului sintactic și semantic în format binar relocabil sau în limbaj de asamblare.

Optimizatorul are rolul de a prelucra ieșirea generatorului de cod cu scopul de a minimiza memoria necesară la execuție și a elimina redundanțele din corpul programului. Unele compilatoare efectuează anumite optimizări înainte de generarea codului.

O funcție importantă a compilatorului constă în detectarea erorilor din programul sursă și corectarea sau acoperirea lor. Mediul de programare (Turbo) Pascal integrează un editor de texte, un compilator, editorul de legături și facilitează depanarea / executarea programelor în limbaj Pascal.

Spre deosebire de compilatoare, interpretoarele efectuează și executarea programului odată cu traducerea programului sursă, furnizând la ieșire rezultatele programului. Mai precis, diferența față de un compilator este aceea că interpretorul nu produce un program obiect ce urmează să fie executat după interpretare, ci chiar execută acest program. Din punct de vedere structural, un interpretor se aseamănă cu un compilator, dar forma intermedieră obținută nu este folosită la generarea de cod, ci pentru a ușura decodificarea instrucțiunii sursă în vederea executării ei. Este cazul interpretorului sistemului AUTOCAD care analizează linia de comandă și, dacă comanda este corectă, realizează funcția solicitată.

Un editor de texte este un program *on-line* (răspunsul la comenzi este imediat), ce acceptă comenzi introduse de la terminal pentru a scrie și/sau șterge caractere, linii sau grupuri de linii din programul sursă sau din orice fișier text [3].

Editoarele de texte pot lucra în mod linie și/sau mod ecran. Ele pun la dispoziție comenzi privind deplasarea cursorului în text (pe linie, în cadrul unui bloc sau în cadrul întregului fișier), manipularea blocurilor de text (marcare, deplasare, copiere, ștergere), comenzi de formatare a ieșirii etc. Exemplificăm prin *Edit* (Ms-DOS, Windows 2000), *Notepad* (Windows), *vi* sau *emacs* (UNIX) etc.

BIBLIOGRAFIE

1. Albeanu G., *Algoritmi și limbaje de programare*. Editura Fundației România de Mâine, București, 2000.
2. Albeanu G., *Programarea și utilizarea calculatoarelor*, Editura Universității din Oradea, 2003.
3. Albeanu G., *Sisteme de operare*. Editura Petrion, 1996.
4. Knuth D.E., *Arta programării calculatoarelor*, Vol. I, Editura Teora, 1999.
5. DEX'1996, *Dicționarul explicativ al limbii române*, Editura Univers Enciclopedic, București, 1996.
6. DINF'1981, *Dicționar de informatică*, Editura Științifică și Enciclopedică, București, 1981.
7. ***, NASM / NdisASM

GEOMETRIE DIFERENȚIALĂ

Prof. univ. dr. I. DUDA

I. TEORIA LOCALĂ A CURBELOR

I.1. Curve în spațiul Euclidian n -dimensional E_n . Tangenta.

Hiperplan normal

Definiție. Considerăm un interval $I \subseteq \mathbb{R}$. Se numește **curbă parametrizată** sau, pe scurt, **curbă** în spațiul E_n , orice aplicație C^∞ -diferențiabilă $c : I \rightarrow E_n$.

Observație. Dacă I nu este interval deschis, atunci aplicația $c : I \rightarrow E_n$ este curbă în E_n dacă există un interval deschis $\bar{I} \subseteq \mathbb{R}$ și o aplicație C^∞ -diferențiabilă $\bar{c} : \bar{I} \rightarrow E_n$ astfel încât $I \subseteq \bar{I}$ și $\bar{c}|_I = c$.

Fie $c : I \rightarrow E_n$ o curbă parametrizată. Punctele care aparțin imaginii $c(I)$ se numesc puncte ale curbei c .

Curbele din E_2 se numesc **curve plane**, iar curbele din E_3 se numesc **curve strâmbă**.

Definiție. Un punct $c(t_0)$ al unei curbe $c : I \rightarrow E_n$ se numește **punct regulat** dacă $\dot{c}(t_0) \neq 0$. Dacă $\dot{c}(t_0) = 0$, atunci punctul $c(t_0)$ al curbei c se numește **punct singular**. Curba $c : I \rightarrow E_n$ se numește **curbă regulată** dacă toate punctele ei sunt puncte regulate, deci dacă $\dot{c}(t) \neq 0$, oricare ar fi $t \in I$.

Observație. Fie M imaginea geometrică a curbei parametrizate $c : I \rightarrow E_n$, deci $M = c(I) \subset E_n$. Se spune că c este **o parametrizare de clasă C^∞ a lui M** .

Considerând în E_n un sistem de coordonate carteziene, o curbă parametrizată este dată prin ecuațiile

$$\begin{cases} x^1 = c^1(t) \\ \dots, \\ x^n = c^n(t) \end{cases}, \quad t \in I, \quad (1.1.)$$

astfel că avem $c(t) = (c^1(t), \dots, c^n(t))$ pentru orice $t \in I$. Se spune că t este **parametrul curbei**. Dacă notăm $P = c(t_1)$, unde $t_1 \in I$, atunci spunem că punctul

P corespunde valorii t_1 a parametrului t (se mai spune că punctul P are **abscisa curbilinie** t_1).

Definiție. Considerăm două curbe parametrizeate $c : I \rightarrow E_n$ și $\bar{c} : \bar{I} \rightarrow E_n$. Se spune că curba \bar{c} a fost obținută din curba c printr-o schimbare de parametru (sau că c și \bar{c} diferă printr-o schimbare de parametru sau că c și \bar{c} sunt echivalente), dacă există un difeomorfism $\varphi : \bar{I} \rightarrow I$ astfel încât

$$\bar{c} = c \circ \varphi. \quad (1.2.)$$

Observație. Din egalitatea (1.2) rezultă $\bar{c}(\bar{I}) = c(I)$, adică curbele parametrizeate c și \bar{c} au aceeași imagine geometrică.

Evident, curbele parametrizeate care diferă între ele printr-o schimbare de parametru formează o clasă de echivalență. O astfel de clasă se numește **curbă neparametrizată**.

Din egalitatea (1.2) rezultă

$$\dot{\bar{c}}(\bar{t}) = \dot{c}(t)\dot{\varphi}(\bar{t}), \text{ unde } t = \varphi(\bar{t}). \quad (1.3.)$$

Egalitatea (1.3) ne arată că vectorii $\dot{\bar{c}}(\bar{t})$ și $\dot{c}(t)$ sunt coliniari. Dacă $\dot{\varphi}(\bar{t}) > 0$, $\forall \bar{t} \in \bar{I}$, atunci se spune că **schimbarea de parametru păstrează orientarea**. În acest caz, egalitatea (1.3) ne arată că vectorii coliniari $\dot{c}(t)$ și $\dot{\bar{c}}(\bar{t})$ au același sens.

Definiție. Considerăm o curbă parametrizată $c : I \rightarrow E_n$ și fie $[a, b] \subseteq I$. Curba parametrizată $c|_{[a, b]} : [a, b] \rightarrow E_n$ se numește **arc** al curbei c . Prin **lungimea arcului** de curbă $c|_{[a, b]}$ înțelegem lungimea imaginii aplicației $c|_{[a, b]}$. Lungimea arcului de curbă $c|_{[a, b]}$ este

$$L\left(c|_{[a, b]}\right) = \int_a^b \|\dot{c}(t)\| dt. \quad (1.4.)$$

Propoziție. Lungimea unui arc de curbă este un invariant al schimbărilor de parametru.

Propoziție. Fie $c : I \rightarrow E_n$ o curbă regulată. Fixăm un punct $t_0 \in I$. Atunci există un interval $J \subset \mathbb{R}$, cu $0 \in J$ și o schimbare de parametru care păstrează orientarea $\varphi : J \rightarrow I$ astfel încât $\varphi(0) = t_0$ și $\left\| \frac{\cdot}{\dot{c} \circ \varphi}(s) \right\| = 1$, $\forall s \in J$.

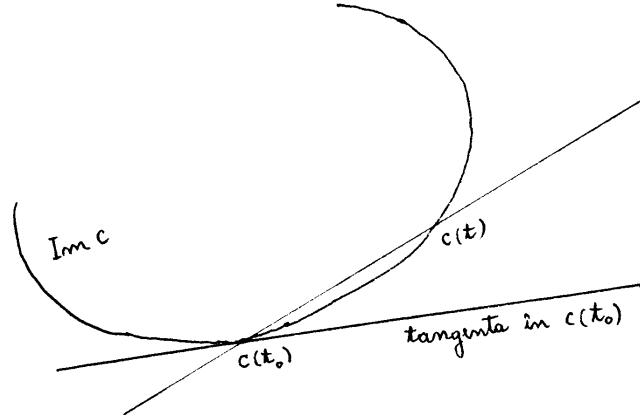
Observație. Propoziția ne arată că orice curbă regulată este echivalentă cu o curbă parametrizată canonic.

Definiție. O curbă $c : I \rightarrow E_n$ se spune să este **parametrizată canonice**, dacă $\|\dot{c}(s)\| = 1$, $\forall s \in I$. Parametrul s este numit **parametrul canonice**.

Fie $c : I \rightarrow E_n$ o curbă parametrizată canonice, deci $\|\dot{c}(s)\| = 1$, $\forall s \in I$. Considerăm un interval $[s_0, s_1] \subset I$. Lungimea arcului de curbă $c|_{[s_0, s_1]}$ este dată

$$\text{de } L\left(c|_{[s_0, s_1]}\right) = \int_{s_0}^{s_1} \|\dot{c}(s)\| ds = \int_{s_0}^{s_1} ds = s_1 - s_0.$$

Definiție. Considerăm o curbă $c : I \rightarrow E_n$ și fie $c(t_0)$ un punct regulat al curbei c . Prin **tangenta** la curba c în punctul $c(t_0)$ înțelegem poziția limită a dreptei determinate de $c(t_0)$ și de un punct oarecare $c(t)$ al curbei când t tinde către t_0 .



Dreapta determinată de punctele $c(t) = (c^1(t), \dots, c^n(t))$ și $c(t_0) = (c^1(t_0), \dots, c^n(t_0))$ are ecuațiile

$$\frac{x^1 - c^1(t_0)}{c^1(t) - c^1(t_0)} = \dots = \frac{x^n - c^n(t_0)}{c^n(t) - c^n(t_0)}.$$

Împărțind numitorii cu $t - t_0$, apoi trecând la limită cu $t \rightarrow t_0$, obținem ecuațiile tangentei la curba c în punctul regulat $c(t_0)$

$$\frac{x^1 - c^1(t_0)}{\dot{c}^1(t_0)} = \dots = \frac{x^n - c^n(t_0)}{\dot{c}^n(t_0)}. \quad (1.5.)$$

Vectorul $\dot{c}(t_0) = (\dot{c}^1(t_0), \dots, \dot{c}^n(t_0))$ este numit vector tangent la curba c în punctul $c(t_0)$.

Definiție. Prin **hiperplan normal** la curba $c : I \rightarrow E_n$ ($n > 2$) în punctul regulat $c(t_0)$ înțelegem hiperplanul care trece prin punctul $c(t_0)$ și este perpendicular pe tangentă la curba c în punctul $c(t_0)$.

Ținând seama de (1.5), obținem ecuația hiperplanului normal la curba $c : I \rightarrow E_n$ ($n > 2$) în punctul $c(t_0)$

$$(x^1 - c^1(t_0))\dot{c}^1(t_0) + \dots + (x^n - c^n(t_0))\dot{c}^n(t_0) = 0. \quad (1.6.)$$

Definiție. Fie $c : I \rightarrow E_2$ o curbă plană. Prin **normală** la curba c în punctul regulat $c(t_0)$ înțelegem dreapta care trece prin punctul $c(t_0)$ și este perpendiculară pe tangentă la curba c în punctul $c(t_0)$.

Exemplu. Fie a și b doi vectori în E_n . Considerăm aplicația $c : \mathbb{R} \rightarrow E_n$ definită prin $c(t) = at + b$. Este clar că c este o curbă parametrizată în E_n . Curba $c : \mathbb{R} \rightarrow E_n$ este regulată dacă și numai dacă $a \neq 0$ și în acest caz curba este o dreaptă.

Propoziție. Fie $c : I \rightarrow E_n$ o curbă regulată. Următoarele afirmații sunt echivalente:

- (i) Hiperplanele normale la curbă trec printr-un punct fix,
- (ii) Imaginea geometrică a curbei parametrizează $c : I \rightarrow E_n$ este situată pe o hipersferă.

I.2. Curve în poziție generală. Hiperplan osculator

Definiție. Fie $c : I \rightarrow E_n$ o curbă parametrizată. Spunem că curba c este în poziție generală dacă vectorii $\dot{c}(t), c^{(2)}(t), \dots, c^{(n-1)}(t)$ sunt liniar independenți, oricare ar fi valoarea parametrului t în intervalul I .

Definiție. Fie $c : I \rightarrow E_n$ ($n \geq 3$) o curbă în poziție generală și fie $t_0 \in I$. Hiperplanul care trece prin punctul $c(t_0)$ și este paralel cu vectorii $\dot{c}(t), c^{(2)}(t), \dots, c^{(n-1)}(t)$ se numește **hiperplan osculator** curbei c în punctul $c(t_0)$.

Observație. Ecuația hiperplanului osculator curbei $c : I \rightarrow E_n$ în punctul $c(t_0) = (c^1(t_0), \dots, c^n(t_0))$ este

$$\begin{vmatrix} x^1 - c^1(t_0) & x^2 - c^2(t_0) & \cdots & x^n - c^n(t_0) \\ \dot{c}^1(t_0) & \dot{c}^2(t_0) & \cdots & \dot{c}^n(t_0) \\ (c^1)^{(2)}(t_0) & (c^2)^{(2)}(t_0) & \cdots & (c^n)^{(2)}(t_0) \\ \vdots & \vdots & \ddots & \vdots \\ (c^1)^{(n-1)}(t_0) & (c^2)^{(n-1)}(t_0) & \cdots & (c^n)^{(n-1)}(t_0) \end{vmatrix} = 0. \quad (2.1.)$$

Propoziție. Fie $c : I \rightarrow E_n$ ($n \geq 3$) o curbă în poziție generală și fie $t_0 \in I$.

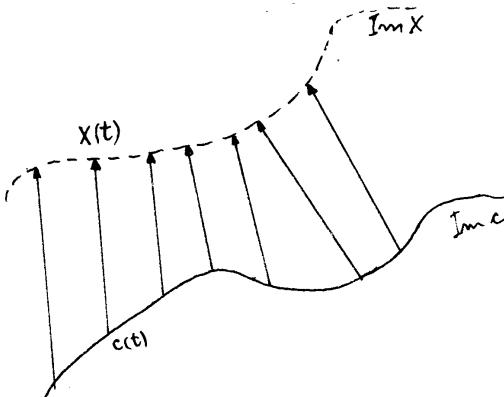
Dacă un hiperplan H ce trece prin punctul $c(t_0)$ intersectează imaginea aplicației c în n puncte confundate în $c(t_0)$, atunci H este hiperplanul osculator curbei în punctul $c(t_0)$.

I.3. Câmpuri de vectori de-a lungul unei curbe. Reperul lui Frenet.

**Teorema de existență și unicitate a reperului Frenet
pentru o curbă în poziție generală**

Definiție. Fie $c : I \rightarrow E_n$ o curbă parametrizată. Prin **câmp de vectori de-a lungul curbei** c înțelegem o aplicație diferențiabilă $X : I \rightarrow E_n$.

În reprezentare geometrică, vectorul $X(t)$ se consideră ca vector tangent la spațiul E_n în punctul $c(t)$, adică se identifică cu vectorul $(c(t), X(t))$ din $T_{c(t)} \times E_n$.



Exemplu. Fie $c : I \rightarrow E_n$ o curbă parametrizată. Atunci aplicația $\dot{c} : t \in I \rightarrow \dot{c}(t) \in E_n \cong T_{c(t)}E_n$ este un câmp vectorial de-a lungul curbei c .

Câmpul vectorial $t \rightarrow \dot{c}(t)$ se numește **câmpul vectorial tangent curbei** $t \rightarrow c(t)$.

Definiție. Fie $c : I \rightarrow E_n$ o curbă în poziție generală. Prin **reper Frenet asociat curbei** c înțelegem un sistem de n câmpuri vectoriale $\{e_1, \dots, e_n\}$ de-a lungul curbei c , astfel încât pentru orice $t \in I$ să avem îndeplinite următoarele proprietăți:

$$(F) \quad \langle e_i(t), e_j(t) \rangle = \delta_{ij}, \quad \forall i, j \in \{1, \dots, n\};$$

$$(F') \quad \text{sp}(\dot{c}(t), c^{(2)}(t), \dots, c^{(k)}(t)) = \text{sp}(e_1(t), \dots, e_k(t)), \quad \forall k \in \{1, \dots, n-1\};$$

(F'') sistemele de vectori $\{\dot{c}(t), c^{(2)}(t), \dots, c^{(k)}(t)\}$ și $\{e_1(t), \dots, e_k(t)\}$ sunt la fel orientate pentru orice $k \in \{1, \dots, n-1\}$;

(F''') sistemul de vectori $\{e_1(t), \dots, e_n(t)\}$ este orientat pozitiv.

Teorema (de existență și unicitate a reperului Frenet pentru o curbă în poziția generală). Fie $c : I \rightarrow E_n$ o curbă în poziție generală. Atunci există un unic reper Frenet $\{e_1, \dots, e_n\}$ asociat curbei c .

I.4. Formulele lui Frenet. Curburile unei curbe. Invarianta curburilor unei curbe la schimbări de parametru ce păstrează orientarea și la izometrii proprii. Teorema fundamentală a teoriei curbelor

Propoziție. Fie $c : I \rightarrow E_n$ o curbă în poziție generală și fie $\{e_1, \dots, e_n\}$ reperul Frenet asociat curbei. Avem formulele

$$\dot{e}_i(t) = \sum_{j=1}^n a_{ij} e_j(t), \quad (4.1.)$$

unde

$$a_{ij}(t) + a_{ji}(t) = 0, \quad \forall i, j \in \{1, \dots, n\} \quad (4.1').$$

și

$$a_{ij}(t) = 0, \text{ dacă } j > i+1. \quad (4.1'').$$

Propoziție. Fie $c : I \rightarrow E_n$ o curbă în poziție generală și fie $\{e_1, \dots, e_n\}$ reperul Frenet asociat curbei. Notăm cu $K_1(t), \dots, K_{n-1}(t)$ curburile curbei într-un punct oarecare $c(t)$ al curbei. Atunci, pentru $n > 2$, avem $K_i(t) > 0$, $\forall i \in \{1, \dots, n-2\}$.

Propoziție. Fie $c : I \rightarrow E_n$ și $\bar{c} : \bar{I} \rightarrow E_n$ două curbe în poziție generală care diferă între ele printr-o schimbare de parametru care păstrează orientarea, adică există un difeomorfism $\varphi : \bar{I} \rightarrow I$ cu $\dot{\varphi}(\bar{t}) > 0$ oricare ar fi $\bar{t} \in \bar{I}$ și astfel încât $\bar{c} = c \circ \varphi$. Dacă $\{e_1, \dots, e_n\}$ este reperul Frenet asociat curbei c și dacă notăm $\bar{e}_i = e_i \circ \varphi$, $i \in \{1, \dots, n\}$, atunci

- i) $\{\bar{e}_1, \dots, \bar{e}_n\}$ este reperul Frenet asociat curbei \bar{c} ;
- ii) $\bar{K}_i(\bar{t}) = K_i(t)$, unde $t = \varphi(\bar{t})$, $i \in \{1, \dots, n-1\}$ și unde $K_i(t)$ și respectiv $\bar{K}_i(\bar{t})$ sunt curburile curbei c , respectiv \bar{c} .

Propoziție. Fie $c : I \rightarrow E_n$ o curbă și $B : E_n \rightarrow E_n$ o izometrie. Atunci:

- i) $\bar{c} = B \circ c : I \rightarrow E_n$ este o curbă parametrizată;
- ii) $\dot{\bar{c}}(t) = R \dot{c}(t)$, unde $R = dB_x$ este componenta ortogonală a izometriei B ;
- iii) $\|\dot{\bar{c}}(t)\| = \|\dot{c}(t)\|$, $\forall t \in I$;
- iv) în cazul în care curba c este în poziție generală, rezultă că și curba \bar{c} este în poziție generală.

Propoziție. Fie $c : I \rightarrow E_n$ o curbă în poziție generală, $\{e_1, \dots, e_n\}$ reperul Frenet asociat curbei și fie $B : E_n \rightarrow E_n$ o izometrie proprie. Notăm $\bar{e}_i = Re_i$, $i \in \{1, \dots, n\}$, unde R este rotația izometriei B . Atunci:

- i) $\{\bar{e}_1, \dots, \bar{e}_n\}$ este reperul Frenet asociat curbei $\bar{c} = B \circ c$;
- ii) $\bar{K}_i(\bar{t}) = K_i(t)$, $\forall i \in \{1, \dots, n-1\}$, $\forall t \in I$.

Propoziție. Fie $c : I \rightarrow E_n$ și $\bar{c} : \bar{I} \rightarrow E_n$ două curbe în poziție generală.

Presupunem că $\forall t \in I$ avem

$$\|\dot{c}(t)\| = \|\dot{\bar{c}}(t)\|, K_i(t) = \bar{K}_i(t), \forall i \in \{1, \dots, n-1\}. \quad (4.2.)$$

Atunci există o unică izometrie proprie $B : E_n \rightarrow E_n$ astfel încât $\bar{c} = B \circ c$.

Teorema (teorema fundamentală a teoriei curbelor). Fie un interval $I \subseteq \mathbb{R}$. Presupunem date $n-1$ funcții diferențiabile $F_i : I \rightarrow \mathbb{R}$, $i \in \{1, \dots, n-1\}$ cu proprietatea că $F_j(s) > 0$, $\forall s \in I$ și $\forall j \in \{1, \dots, n-2\}$. Atunci există o curbă parametrizată $c : s \in I \rightarrow c(s) \in E_n$, $\|\dot{c}(s)\| = 1$, $\forall s \in I$, unică, abstracție făcând de o izometrie proprie, ale cărei curburi sunt $K_i(s) = F_i(s)$, $\forall i \in \{1, \dots, n-1\}$.

Propoziție. Fie $c : I \rightarrow E_n$ o curbă în poziție generală. Următoarele afirmații sunt echivalente:

- (i) ultima curbură a curbei c este nulă
- (ii) imaginea aplicației c este inclusă într-un hiperplan.

Exemplu. Ne propunem să determinăm curbele din E_n care au curburile constante. Fie $c : I \rightarrow E_n$ o curbă în poziție generală. Presupunem că curba este canonic parametrizată, deci $\|\dot{c}(s)\| = 1$, $\forall s \in I$. Avem formulele Frenet

$$e_1(s) = \dot{c}(s), \quad \dot{e}_1(s) = \sum_{j=1}^n a_{ij}(s) e_j(s), \quad (4.3.)$$

unde

$$a_{ij}(t) + a_{ji}(t) = 0, \quad a_{ij}(s) = 0 \text{ dacă } j > i+1. \quad (4.3').$$

Curburile curbei c sunt:

$$K_i(s) = a_{i,i+1}(s), \quad i \in \{1, \dots, n-1\}. \quad (4.4.)$$

Presupunem în continuare că $K_i(s) = K_i = \text{constant}$, oricare ar fi $i \in \{1, \dots, n-1\}$. Dacă notăm $e_i = (e_i^1, \dots, e_i^n)$, atunci din (4.2), avem

$$\dot{e}_i^k(s) = \sum_{j=1}^n a_{ij}(s) e_j^k(s), \quad k \in \{1, \dots, n\}. \quad (4.5.)$$

Dacă notăm $\phi = (e_j^i)_{1 \leq i,j \leq n}$, atunci (4.5.) se scrie:

$$\frac{d\phi(s)}{ds} = a\phi(s), \quad (4.6.)$$

unde $a = (a_{ij})$ este o matrice constantă.

Alegem axele astfel încât pentru $s = 0$, versorii $e_i(0)$ să devină versorii axelor Ox_i , deci presupunem condiția inițială de forma

$$\phi(0) = E = \begin{vmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{vmatrix}. \quad (4.7.)$$

Soluția ecuației (4.6), (4.7) este

$$\phi(s) = E + \frac{s}{1!} a + \frac{s^2}{2!} a^2 + \dots \quad (4.8.)$$

Ca să determinăm curba va trebui să luăm prima linie din matricea ϕ . Va rezulta vectorul $e_1(s)$. Dar $e_1(s) = \dot{c}(s)$ și prin integrare obținem $c(s)$. Să

facem calculul efectiv în cazul $n = 2$. Avem $a = \begin{vmatrix} 0 & K_1 \\ -K_1 & 0 \end{vmatrix}$,

$$a^2 = \begin{vmatrix} -K_1^2 & 0 \\ 0 & -K_1^2 \end{vmatrix}, \quad a^3 = \begin{vmatrix} 0 & -K_1^3 \\ K_1^3 & 0 \end{vmatrix}, \quad a^4 = \begin{vmatrix} K_1^4 & 0 \\ 0 & K_1^4 \end{vmatrix}, \dots \quad \text{Matricea } \phi \text{ din}$$

(4.8) se scrie:

$$\phi = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} + \frac{s}{1!} \begin{vmatrix} 0 & K_1 \\ -K_1 & 0 \end{vmatrix} + \frac{s^2}{2!} \begin{vmatrix} -K_1^2 & 0 \\ 0 & -K_1^2 \end{vmatrix} + \frac{s^3}{3!} \begin{vmatrix} 0 & -K_1^3 \\ K_1^3 & 0 \end{vmatrix} + \frac{s^4}{4!} \begin{vmatrix} K_1^4 & 0 \\ 0 & K_1^4 \end{vmatrix} + \dots$$

sau

$$\phi = \begin{vmatrix} 1 - \frac{(sK_1)^2}{2!} + \frac{(sK_1)^4}{4!} - \frac{(sK_1)^6}{6!} + \dots & sK_1 - \frac{(sK_1)^3}{3!} + \frac{(sK_1)^5}{5!} - \frac{(sK_1)^7}{7!} + \dots \\ \dots & \dots \end{vmatrix}$$

sau

$$\phi = \begin{vmatrix} \cos sK_1 & \sin sK_1 \\ \dots & \dots \end{vmatrix}.$$

Deoarece prima linie din matricea ϕ reprezintă componentele vectorului $e_1(s)$, avem $e_1(s) = (\cos sK_1, \sin sK_1)$. Deoarece $e_1(s) = \dot{c}(s) = (\dot{x}(s), \dot{y}(s))$, rezultă ușor $c(s) = (x(s), y(s))$ din ecuațiile

$$\dot{x}(s) = \cos sK_1, \quad \dot{y}(s) = \sin sK_1. \quad (4.9.)$$

Presupunem $K_1 = 0$. Atunci din (4.9) rezultă $x(s) = s + a$, $y(s) = b$, a și b constante și deci curba este o dreaptă și anume dreapta $y = b (= \text{constant})$.

Prin urmare, curbele plane care au curbura nulă sunt drepte.

Dacă $K_1 \neq 0$, atunci din (4.9) obținem $\begin{cases} x(s) = \frac{1}{K_1} \sin sK_1 + a \\ y(s) = -\frac{1}{K_1} \cos sK_1 + b \end{cases}$, unde a

și b sunt constante și deci curba este un cerc și anume cercul $(x-a)^2 + (y-b)^2 = \frac{1}{K_1^2}$. Prin urmare curbele plane care au curbura constantă (nenuată) sunt cercuri.

I.5. Curve în spațiul euclidian tridimensional E_3 (Curve strâmbă)

Expresia curburii și torsioniunii unei曲be strâmbă intr-o parametrizare arbitrară

Propoziție. Considerăm o curbă în poziție generală $c: I \rightarrow E_3$, $t \rightarrow c(t) = (x(t), y(t), z(t))$. Fie $K_1(t)$ și $K_2(t)$ curburile curbei c într-un punct oarecare $c(t)$. Atunci avem formulele:

$$K_1(t) = \frac{\|\dot{c}(t) \times c^{(2)}(t)\|}{\|\dot{c}(t)\|^3}, \quad (5.1.)$$

$$K_2(t) = \frac{\det(\dot{c}(t), c^{(2)}(t), c^{(3)}(t))}{\|\dot{c}(t) \times c^{(2)}(t)\|^2}, \quad (5.2.)$$

$$K_1(t) = \frac{(A^2(t) + B^2(t) + C^2(t))^{\frac{1}{2}}}{(\dot{x}^2(t) + \dot{y}^2(t) + \dot{z}^2(t))^{\frac{3}{2}}}, \quad (5.1').$$

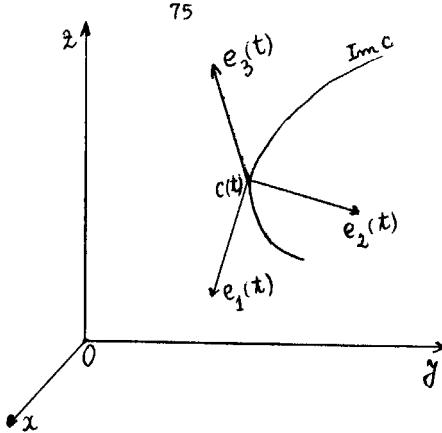
$$K_2(t) = \frac{\begin{vmatrix} \dot{x}(t) & \dot{y}(t) & \dot{z}(t) \\ \ddot{x}(t) & \ddot{y}(t) & \ddot{z}(t) \\ \ddot{\ddot{x}}(t) & \ddot{\ddot{y}}(t) & \ddot{\ddot{z}}(t) \end{vmatrix}}{A^2(t) + B^2(t) + C^2(t)}, \quad (5.2').$$

unde am folosit notațiile:

$$A(t) = \begin{vmatrix} \dot{y}(t) & \dot{z}(t) \\ \ddot{y}(t) & \ddot{z}(t) \end{vmatrix}, \quad B(t) = \begin{vmatrix} \dot{z}(t) & \dot{x}(t) \\ \ddot{z}(t) & \ddot{x}(t) \end{vmatrix}, \quad C(t) = \begin{vmatrix} \dot{x}(t) & \dot{y}(t) \\ \ddot{x}(t) & \ddot{y}(t) \end{vmatrix}.$$

Triedrul lui Frenet

Fie $c : I \rightarrow E_3$ o curbă în poziție generală și fie $\{e_1(t), e_2(t), e_3(t)\}$ reperul lui Frenet într-un punct $c(t)$ al curbei



Dreapta situată la intersecția planului normal cu planul osculator curbei într-un punct $c(t)$ se numește **normală principală**, iar dreapta perpendiculară pe planul osculator curbei în punctul $c(t)$ se numește **binormală** la curbă în acel punct. Planul determinat de tangenta și binormala la curbă într-un punct $c(t)$ se numește **planul rectificat** al curbei. În definitiv, în fiecare punct $c(t)$ al unei curbe strâmbă în poziție generală putem să asociem un triedru tridreptunghic (numit **triедрul lui Frenet**) ale căruia muchii sunt tangenta, normala principală și binormala și ale cărui fețe sunt planul normal, planul osculator și planul rectificant.

Fie $c : t \in I \rightarrow c(t) = (x(t), y(t), z(t)) \in E_3$ o curbă în poziție generală. Obținem fără dificultate ecuațiile muchiilor și fețelor triedrului Frenet asociat curbei într-un punct $c(t) = (x(t), y(t), z(t))$ și anume

- ecuațiile tangentei

$$\frac{x - x(t)}{\dot{x}(t)} = \frac{y - y(t)}{\dot{y}(t)} = \frac{z - z(t)}{\dot{z}(t)},$$

- ecuația planului normal

$$\dot{x}(t)(x - x(t)) + \dot{y}(t)(y - y(t)) + \dot{z}(t)(z - z(t)) = 0,$$

- ecuația planului osculator

$$A(t)(x - x(t)) + B(t)(y - y(t)) + C(t)(z - z(t)) = 0,$$

unde

$$A(t) = \begin{vmatrix} \dot{y}(t) & \dot{z}(t) \\ \ddot{y}(t) & \ddot{z}(t) \end{vmatrix}, \quad B(t) = \begin{vmatrix} \dot{z}(t) & \dot{x}(t) \\ \ddot{z}(t) & \ddot{x}(t) \end{vmatrix}, \quad C(t) = \begin{vmatrix} \dot{x}(t) & \dot{y}(t) \\ \ddot{x}(t) & \ddot{y}(t) \end{vmatrix},$$

- ecuațiile binormalei

$$\frac{x - x(t)}{A(t)} = \frac{y - y(t)}{B(t)} = \frac{z - z(t)}{C(t)},$$

- ecuațiile normalei principale

$$\frac{x - x(t)}{l(t)} = \frac{y - y(t)}{m(t)} = \frac{z - z(t)}{n(t)},$$

unde

$$l(t) = \begin{vmatrix} \dot{y}(t) & \dot{z}(t) \\ B(t) & C(t) \end{vmatrix}, \quad m(t) = \begin{vmatrix} \dot{z}(t) & \dot{x}(t) \\ C(t) & A(t) \end{vmatrix}, \quad n(t) = \begin{vmatrix} \dot{x}(t) & \dot{y}(t) \\ A(t) & B(t) \end{vmatrix},$$

- ecuația planului rectificant

$$l(t)(x - x(t)) + m(t)(y - y(t)) + n(t)(z - z(t)) = 0.$$

Teorema (Lancret). Fie $c : I \rightarrow E_3$ o curbă în poziție generală și fie $K_1(t)$

și $K_2(t)$ curburile curbei într-un punct oarecare $c(t)$. Presupunem că $K_2(t) \neq 0$, $\forall t \in I$. Următoare afirmații sunt echivalente:

- tangenta în fiecare punct al curbei formează un unghi constant cu o direcție fixă;
- normala principală în fiecare punct al curbei este perpendiculară pe o direcție fixă;
- binormala în orice punct al curbei formează un unghi constant cu o direcție fixă;
- curbura și torsiunea curbei diferă printr-un factor constant.

Cerc și sferă osculatoare la o curbă strâmbă

Definiție. Considerăm o curbă în poziție generală $c : I \rightarrow E_3$ și fie M_0 un punct al curbei. Cercul din planul osculator la curbă în M_0 , care întâlnește imaginea aplicației c în trei puncte confundante în acest punct, se numește **cercul osculator** curbei în punctul M_0 .

Propoziție. Considerăm o curbă în poziție generală $c : s \in I \rightarrow c(s) \in E_3$, $\|\dot{c}(s)\| = 1$, $\forall s \in I$ și fie $c(s_0) = M_0$ un punct al curbei. Centrul cercului osculator curbei în punctul M_0 este situat pe normala principală la curbă în acest punct, iar raza cercului osculator este egală cu $\frac{1}{K_1(s_0)}$, unde $K_1(s_0)$ este curbura curbei c în punctul M_0 .

Observație. Cercul osculator unei curbe strâmbă într-un punct se mai numește **cerc de curbură** al curbei în punctul respectiv, iar centrul lui se numește **centrul de curbură**.

Definiție. Considerăm o curbă în poziție generală $c : I \rightarrow E_3$ și fie M_0 un punct al curbei. Sfera care intersectează imaginea aplicației c în patru puncte confundante în M_0 se numește **sferă osculatoare** curbei în punctul M_0 .

Propoziție. Considerăm o curbă în poziție generală $c : s \in I \rightarrow c(s) = (x^1(s), x^2(s), x^3(s)) \in E_3$, $\|\dot{c}(s)\| = 1$, $\forall s \in I$. Fie $c(s_0)$ un punct al curbei. Presupunem că torsionarea curbei nu se anulează în punctul $c(s_0)$. Coordonatele a^1, a^2, a^3 ale centrului sferei osculatoare curbei în punctul $c(s_0) = (x^1(s_0), x^2(s_0), x^3(s_0))$ sunt date de:

$$a^i = x^i(s_0) + R(s_0)e_2^i(s_0) + \dot{R}(s_0)T(s_0)e_3^i(s_0), \quad i \in \{1, 2, 3\},$$

iar raza r a sferei osculatoare curbei în $c(s_0)$ este

$$r = \sqrt{R^2(s_0) + \dot{R}^2(s_0)T^2(s_0)}, \quad \text{unde am folosit notațiile } R(s_0) = \frac{1}{K_1(s_0)},$$

$$T(s_0) = \frac{1}{K_2(s_0)} (K_1(s) \text{ și } K_2(s) \text{ sunt curburile curbei } c \text{ în punctul } c(s)).$$

I.6. Curbe plane

Curbura unei curbe din E_2 . Formulele Frenet.

Interpretarea geometrică a curburii unei curbe plane

Fie $c : I \rightarrow E_2$ o curbă plană regulată. Evident, curba c este în poziție generală. Notăm cu $\{e_1, e_2\}$ reperul Frenet asociat curbei.

Prima curbă $K_1(t)$ a curbei c în punctul $c(t)$ se mai numește **curbura curbei c în punctul $c(t)$** .

Propoziție. Considerăm o curbă regulată $c : t \in I \rightarrow c(t) = (x(t), y(t)) \in E_2$. Avem formulele:

$$K_1(t) = \frac{\det(\dot{c}(t), \ddot{c}(t))}{\|\dot{c}(t)\|^3}, \quad (6.1.)$$

$$K_1(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{\frac{3}{2}}}. \quad (6.1').$$

Cerc osculator

Definiție. Considerăm o curbă $c : I \rightarrow E_2$ și fie $t_0 \in I$ astfel încât $\dot{c}(t) \neq 0$ și $K_1(t_0) \neq 0$. Un cerc se numește **cerc osculator** curbei în punctul $c(t_0)$, dacă are cu imaginea curbei c trei puncte confundate în $c(t_0)$.

Propoziție. Considerăm o curba parametrizată $c : t \in I \rightarrow c(t) = (x(t), y(t)) \in E_2$ și fie $t_0 \in I$ astfel încât $\dot{c}(t) \neq 0$ și $K_1(t_0) \neq 0$. Cercul osculator curbei în punctul $c(t_0)$ are centrul în punctul $\Omega_0 = (a, b)$ și raza $R(t_0)$, unde

$$a = x(t_0) + \dot{y}(t_0) \frac{\dot{x}^2(t_0) + \dot{y}^2(t_0)}{\ddot{x}(t_0)\dot{y}(t_0) - \dot{x}(t_0)\ddot{y}(t_0)}, \quad (6.2.)$$

$$b = y(t_0) - \dot{x}(t_0) \frac{\dot{x}^2(t_0) + \dot{y}^2(t_0)}{\ddot{x}(t_0)\dot{y}(t_0) - \dot{x}(t_0)\ddot{y}(t_0)},$$

$$R(t_0) = \frac{1}{|K_1(t_0)|}, \quad (6.3.)$$

Centrul cercului osculator curbei în punctul $c(t_0)$ se numește **centrul de curbură** al curbei c în punctul $c(t_0)$, iar raza $R(t_0)$ se numește **raza de curbură** a curbei c în punctul $c(t_0)$.

Evoluta unei curbe plane

Fie $c : t \in I \rightarrow c(t) = (x(t), y(t)) \in E_2$ o curbă regulată cu proprietatea că pentru orice $t \in I$, $K_1(t) \neq 0$. Atunci cercul osculator curbei c într-un punct oarecare $c(t)$ are centrul $\Omega = (X(t), Y(t))$ unde

$$\begin{cases} X(t) = x(t) - \frac{\dot{y}(t)(\dot{x}^2(t) + \dot{y}^2(t))}{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)} \\ Y(t) = y(t) + \frac{\dot{x}(t)(\dot{x}^2(t) + \dot{y}^2(t))}{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)} \end{cases} \quad (6.4.)$$

Curba $\Gamma : t \in I \rightarrow \Gamma(t) = (X(t), Y(t)) \in E_2$ se numește **evolută curbei** c .

II. ELEMENTE DE TEORIA GLOBALĂ A CURBELOR

II.1. Curve simple. Curve închise. Inegalitatea izoperimetrică

Definiție. O curbă $c : [a, b] \rightarrow E_2$ se numește curbă simplă dacă pentru orice $t_1, t_2 \in [a, b]$ cu $t_1 \neq t_2$ avem $c(t_1) \neq c(t_2)$.

Exemple

1. **Elipsa** $c : [0, 2\pi] \rightarrow E_2$, $c(t) = (a \cos t, b \sin t)$, $a > b > 0$ este o curbă simplă (Fig. 1.).
2. **Astroida** $c : [0, 2\pi] \rightarrow E_2$, $c(t) = (r \cos^3 t, r \sin^3 t)$, $r > 0$ este o curbă simplă (Fig. 2).

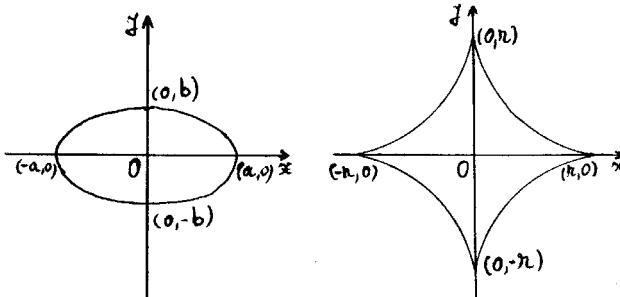
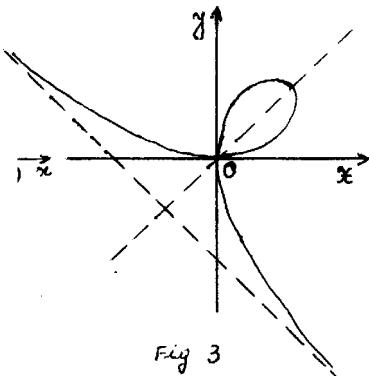


Fig. 1

Fig. 2

3. **Foliul lui Descartes** $c : \mathbb{R} \setminus \{-1\} \rightarrow E_2$, $c(t) = \left(\frac{3at}{1+t^3}, \frac{3at^2}{1+t^3} \right)$, $a > 0$ este o curbă simplă (Fig. 3).



4. **Spirala lui Arhimede** $c : [0, 2\pi] \rightarrow E_2$, $c(t) = (at \cos t, at \sin t)$, $a > 0$ este o curbă simplă.
5. **Strofoida** $c : \mathbb{R} \rightarrow E_2$, $c(t) = \left(a \frac{t^2 - 1}{t^2 + 1}, at \frac{t^2 - 1}{t^2 + 1} \right)$, $a > 0$ nu este o curbă simplă (avem $c(1) = c(-1) = (0, 0)$).
6. **Melcul lui Pascal** $c : [0, 2\pi] \rightarrow E_2$, $c(t) = ((1 + 2 \cos t) \cos t, (1 + 2 \cos t) \sin t)$ nu este o curbă simplă (avem $c\left(\frac{2\pi}{3}\right) = c\left(\frac{4\pi}{3}\right) = (0, 0)$).

Definiție. O curbă $c : [a, b] \rightarrow E_2$ se numește **curbă închisă** dacă există o curbă $\bar{c} : \mathbb{R} \rightarrow E_2$ astfel încât $\bar{c}|_{[a, b]} = c$ și $\bar{c}(t+T) = \bar{c}(t)$, $\forall t \in \mathbb{R}$, unde $T = b - a$. Numărul T este **perioada** lui \bar{c} . Cura \bar{c} se numește **periodică cu perioada** T .

Dându-se o curbă închisă c , i se poate asocia o curbă periodică \bar{c} în mod unic.

O definiție echivalentă a curbelor închise este următoarea.

Definiție. O curbă $c : [a, b] \rightarrow E_2$ se numește curbă închisă dacă $c(a) = c(b)$ și $c^{(i)}(a) = c^{(i)}(b)$ pentru orice $i = 1, 2, \dots$

Exemple

1. **Cercul** $c : [0, 2\pi] \rightarrow E_2$, $c(t) = (r \cos t, r \sin t)$, $r > 0$ este o curbă închisă.
2. **Elipsa** este o curbă închisă.
3. **Foliul lui Descartes** și **spirala lui Arhimede** nu sunt curbe închise.

Lema. Considerăm o curbă plană regulată, simplă și închisă $c : [a, b] \rightarrow E_2$, $c(s) = (x(s), y(s))$ și fie S aria domeniului D mărginit de imaginea aplicației c . Atunci avem

$$S = \int_a^b x(s) \dot{y}(s) ds = - \int_a^b \dot{x}(s) y(s) ds. \quad (1.1.)$$

Teorema ce urmează își propune să dea răspuns la următoarea întrebare: dintre toate curbele plane regulate, simple și închise, având aceeași lungime L , care mărginește domeniul cu aria maximă?

Teorema. Considerăm o curbă plană regulată, simplă și închisă, având lungimea L și fie S aria domeniului D mărginit de curba c . Atunci avem

$$4\pi S \leq L^2. \quad (1.2.)$$

Semnul egal are loc dacă și numai dacă curba c este un cerc. Inegalitatea (1.2) poartă numele de **inegalitatea izoperimetrică**.

II.2. Curve ovale. Teorema lui Herglotz

Definiție. O curbă $c : I \rightarrow E_2$ se numește **curbă convexă**, dacă orice punct $c(t)$ al curbei are următoarea proprietate că toate punctele curbei (exceptând punctul $c(t)$) se află în unul din cele două semiplane determinate de tangenta la curbă în punctul $c(t)$.

Definiție. Fie $c : I \rightarrow E_2$ o curbă regulată și $t_0 \in I \subset I$ un punct interior intervalului I . Spunem că $c(t_0)$ este **vârf** al curbei c dacă $K_1(t_0) = 0$.

Definiție. Fie c o curbă regulată având curbura pozitivă în fiecare punct. Se spune că c este **curbă ovală**, dacă ea este simplă, închisă și convexă.

Teorema (Herglotz). O curbă ovală are cel puțin patru vârfuri.

III. Teoria hipersuprafețelor

III.1. Hipersuprafețe în E_{n+1} . Hiperplan tangent normală la o hipersuprafață

Definiție. Fie U o mulțime deschisă în \mathbb{R}^n . Prin **hipersuprafață parametrizată** sau pe scurt **hipersuprafață** în E_{n+1} înțelegem o aplicație diferențiabilă $f : U \rightarrow E_{n+1}$ astfel încât aplicația liniară $df_x : \mathbb{R}^n \cong T_x \mathbb{R}^n \rightarrow \mathbb{R}^{n+1} \cong T_{f(x)} \mathbb{R}^{n+1}$ este injectivă, oricare ar fi punctul $x = (x^1, x^2, \dots, x^n) \in U$.

Se spune că x^1, x^2, \dots, x^n sunt **parametrii hipersuprafeței**.

Hipersuprafețele din E_3 se numesc **suprafețe**.

Definiție. Fie U și \bar{U} două mulțimi deschise în \mathbb{R}^n și fie $f: U \rightarrow E_{n+1}$, $\bar{f}: \bar{U} \rightarrow E_{n+1}$ două hipersuprafețe parametrizate. Spunem că f și \bar{f} diferă printr-o schimbare de parametri, dacă există un difeomorfism $\varphi: \bar{U} \rightarrow U$, astfel încât $\bar{f} = f \circ \varphi$.

Este evident că dacă hipersuprafețele f și \bar{f} diferă printr-o schimbare de parametri, atunci imaginile aplicațiilor f și \bar{f} coincid, deci $f(U) = \bar{f}(\bar{U})$.

Este ușor de văzut că hipersuprafețele care diferă una de alta printr-o schimbare de parametri formează o clasă de echivalență. O astfel de clasă de echivalență se numește **hipersuprafață neparametrizată**.

Definiție. Se spune că schimbarea de parametri păstrează orientarea dacă determinantul funcțional $\left| \frac{\partial \varphi^i(\bar{x})}{\partial \bar{x}^j} \right|_{1 \leq i, j \leq n}$, $\varphi = (\varphi^1, \varphi^2, \dots, \varphi^n)$ este pozitiv.

Definiție. Prin **hiperplan tangent la hipersuprafață** $f: U \rightarrow E_{n+1}$ în punctul $f(x) = (f^1(x), f^2(x), \dots, f^{n+1}(x))$ înțelegem hiperplanul care trece prin punctul $f(x)$ și este paralel cu vectorii $f_{x^1}(x), \dots, f_{x^n}(x)$.

Hiperplanul tangent la hipersuprafeței $f: U \rightarrow E_{n+1}$ în punctul $f(x) = (f^1(x), f^2(x), \dots, f^{n+1}(x))$ are ecuația

$$\begin{vmatrix} x^1 - f^1(x) & \cdots & x^{n+1} - f^{n+1}(x) \\ f_{x^1}^1(x) & \cdots & f_{x^{n+1}}^{n+1}(x) \\ \vdots & \ddots & \vdots \\ f_{x^n}^1(x) & \cdots & f_{x^{n+1}}^{n+1}(x) \end{vmatrix} = 0. \quad (1.1.)$$

Definiție. Dreapta care trece prin punctul $f(x)$ și este perpendiculară pe hiperplanul tangent la hipersuprafeței în punctul $f(x)$ se numește **normală la hipersuprafață**.

Fie $I \subset \mathbb{R}$ un interval deschis. Considerăm aplicația $f: I \times \mathbb{R} \rightarrow E_3$ definită prin

$$f(x^1, x^2) = (X_0(x^1) + x^2 l(x^1), Y_0(x^1) + x^2 m(x^1), Z_0(x^1) + n(x^1)), \quad (1.2.)$$

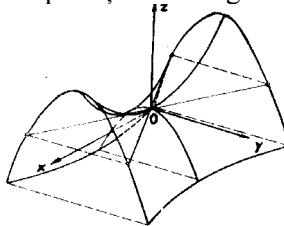
unde funcțiile X_0, Y_0, Z_0, l, m, n depind diferențialibl de $x^1 \in I$ și $l^2(x^1) + m^2(x^1) + n^2(x^1) > 0$. Este ușor de văzut că imaginea aplicației f este generată de o dreaptă care trece printr-un punct oarecare al curbei strâmbă $c : x^1 \in I \rightarrow c(x^1) = (X_0(x^1), Y_0(x^1), Z_0(x^1)) \in E_3$ și are parametrii directori $l(x^1), m(x^1), n(x^1)$.

Aplicația $f : I \times \mathbb{R} \rightarrow E_3$ definită prin (1.2) este o imersie, deci o suprafață parametrizată în E_3 . Această suprafață se numește **suprafață riglată**. Dreapta variabilă d_{x^1} se numește **generatoarea suprafeței**, iar curba c se numește **curbă directoare**.

Exemplu. Este ușor de verificat că aplicația $f : \mathbb{R}^2 \rightarrow E_3$ definită prin $f(x^1, x^2) = (a(x^1 + x^2), b(x^2 - x^1), 2x^1 x^2)$, $a > 0$, $b > 0$ este o imersie.

Imaginea aplicației f este paraboloidul hiperbolic de ecuație $\frac{X^2}{a^2} - \frac{Y^2}{b^2} = 2Z$.

Paraboloidul hiperbolic este o suprafață dublu riglată.



Definiție. O suprafață riglată se numește **suprafață desfășurabilă**, dacă planul tangent la suprafață în punctele unei generatoare este același.

Planul tangent într-un punct $f(x)$ la suprafață riglată considerată mai sus are ecuația

$$\begin{vmatrix} X - X_0(x^1) - x^2 l(x^1) & Y - Y_0(x^1) - x^2 m(x^1) & Z - Z_0(x^1) - x^2 n(x^1) \\ X'_0(x^1) + x^2 l'(x^1) & Y'_0(x^1) + x^2 m'(x^1) & Z'_0(x^1) + n'(x^1) \\ l(x^1) & m(x^1) & n(x^1) \end{vmatrix} = 0$$

Dacă adunăm la prima linie elementele liniei a treia înmulțite cu x^2 , ecuația planului tangent la suprafață în punctul $f(x)$ devine

$$\begin{vmatrix} X - X_0(x^1) & Y - Y_0(x^1) & Z - Z_0(x^1) \\ X'_0(x^1) + x^2 l'(x^1) & Y'_0(x^1) + x^2 m'(x^1) & Z'_0(x^1) + n'(x^1) \\ l(x^1) & m(x^1) & n(x^1) \end{vmatrix} = 0. \quad (1.3.)$$

III.2. Câmpuri de vectori tangenți unei hipersuprafețe. Reper Gauss

Spațiul tangent într-un punct la o hipersuprafață

Fie U o mulțime deschisă în \mathbb{R}^n și fie $f: U \rightarrow E_{n+1}$ o hipersuprafață, deci aplicația liniară $df_x: \mathbb{R}^n \cong T_x \mathbb{R}^n \rightarrow \mathbb{R}^{n+1} \cong T_{f(x)} \mathbb{R}^{n+1}$ este injectivă, oricare ar fi $x \in U$. Notăm $T_{f(x)} f = df_x(T_x \mathbb{R}^n)$. Deoarece aplicația df_x este injectivă, rezultă că $T_{f(x)} f$ este subspațiu vectorial al spațiului $\mathbb{R}^{n+1} \cong T_{f(x)} \mathbb{R}^{n+1}$. Spațiul vectorial $T_{f(x)} f$ are dimensiunea n .

Definiție. *Spațiul vectorial $T_{f(x)} f$ se numește **spațiu tangent la hipersuprafață f în punctul $f(x)$** . Elementele spațiului tangent $T_{f(x)} f$ se numesc **vectori tangenți în punctul $f(x)$ la hipersuprafață f** .*

Câmpuri de vectori de-a lungul unei hipersuprafețe

Definiție. *Prin **câmp de vectori de-a lungul hipersuprafeței** $f: U \rightarrow E_{n+1}$ înțelegem o aplicație diferențierabilă $X: U \rightarrow E_{n+1}$ astfel încât $X(x) \equiv (f(x), X(x)) \in T_{f(x)} E_{n+1} \cong E_{n+1}$, $\forall x \in U$.*

Câmpuri de vectori tangenți unei hipersuprafețe

Definiție. *Fie $X: U \rightarrow E_{n+1}$ un câmp de vectori de-a lungul hipersuprafeței $f: U \rightarrow E_{n+1}$. X se numește **câmp de vectori tangenți hipersuprafeței f** dacă $X(x) \in T_{f(x)} f$, $\forall x \in U$.*

Propoziție. Fie $X : U \rightarrow E_{n+1}$ un câmp de vectori tangenți hipersuprafeței $f : U \rightarrow E_{n+1}$. Atunci există și sunt unice aplicațiile diferențiabile $X^i : U \rightarrow \mathbb{R}$, $i \in \{1, \dots, n\}$, astfel încât să avem

$$X(x) = X^i(x) f_{x^i}(x), \quad \forall x \in U. \quad (2.1.)$$

Câmpuri vectoriale normale

Definiție. Fie $X : U \rightarrow E_{n+1}$ un câmp de vectori de-a lungul hipersuprafeței $f : U \rightarrow E_{n+1}$. X se numește **câmp vectorial normal** hipersuprafeței, dacă oricare ar fi $x \in U$, vectorul $X(x)$ este ortogonal spațiului $T_{f(x)}f$, adică $\langle X(x), f_{x^i}(x) \rangle = 0$, $\forall x \in U$, $\forall i \in \{1, \dots, n\}$.

Reperul lui Gauss

Fie $X : U \rightarrow E_{n+1}$ o hipersuprafață parametrizată și fie $\{f_{x^1}(x), \dots, f_{x^n}(x)\}$ baza canonica a spațiului tangent $T_{f(x)}f$. Notăm

$$N(x) = \frac{f_{x^1}(x) \times \dots \times f_{x^n}(x)}{\|f_{x^1}(x) \times \dots \times f_{x^n}(x)\|}. \quad (2.2.)$$

Este evident că $x \rightarrow N(x)$ este câmp vectorial unitar normal hipersuprafeței f . Reperul $\{f_{x^1}(x), \dots, f_{x^n}(x), N(x)\}$ se numește **reper Gauss** în punctul $f(x)$.

Aplicația $N : U \rightarrow E_{n+1}$ unde $N(x)$ este dat de (2.2), se numește **aplicația Gauss**. Este ușor de văzut că imaginea aplicației Gauss este inclusă în sfera unitate $S_n = \{v \in E_{n+1} \mid \|v\| = 1\}$.

III.3. Prima formă fundamentală a unei hipersuprafețe.

**Invarianța primei forme fundamentale la schimbări de parametri
și la izometrii ale spațiului euclidian E_{n+1}**

Definiție. Fie V un spațiu vectorial real cu n dimensiuni. Se numește **formă biliniară simetrică** orice aplicație $g : V \times V \rightarrow \mathbb{R}$ care îndeplinește condițiile $g(X, Y) = g(Y, X)$, $g(aX + bY, Z) = ag(X, Z) + bg(Y, Z)$, pentru orice vectori $X, Y, Z \in V$ și orice $a, b \in \mathbb{R}$.

Definiție. Spunem că g este **pozitiv definită** dacă pentru orice $X \in V$, $X \neq 0$ avem $g(X, X) > 0$.

Definiție. Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață. Atunci pentru orice $x \in U$ are loc incluziunea $T_{f(x)}f \subset T_{f(x)}E_{n+1} \cong E_{n+1}$. Prin această incluziune este dată o formă biliniară simetrică $g_x : T_{f(x)}f \times T_{f(x)}f \rightarrow \mathbb{R}$, indusă de produsul scalar din E_{n+1} , deci

$$g_x(X, Y) = \langle X, Y \rangle, \quad \forall X, Y \in T_{f(x)}f. \quad (3.1.)$$

Aplicația $x \rightarrow g_x$ se numește **prima formă fundamentală a hipersuprafeței**.

Definiție. Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață. Prin injecția liniară $df_x : \mathbb{R}^n \cong T_x\mathbb{R}^n \rightarrow \mathbb{R}^{n+1} \cong T_{f(x)}\mathbb{R}^{n+1}$ este dată o formă biliniară simetrică $I_x : T_x\mathbb{R}^n \times T_x\mathbb{R}^n \rightarrow \mathbb{R}$ indusă de produsul scalar din $E_{n+1} \cong T_{f(x)}E_{n+1}$, deci

$$I_x(X, Y) = \langle df_x X, df_x Y \rangle, \quad \forall X, Y \in T_x\mathbb{R}^n. \quad (3.1').$$

Aplicația $x \rightarrow I_x$ se numește tot **prima formă fundamentală a hipersuprafeței**.

Propoziție. Fie $(g_{ij}(x))$ matricea formei biliniare simetrice g_x , relativă la baza canonica $\{f_{x^1}(x), \dots, f_{x^n}(x)\}$ a spațiului tangent $T_{f(x)}f$. Atunci:

- i) funcțiile $g_{ij} : U \rightarrow \mathbb{R}$, $x \rightarrow g_{ij}(x)$ sunt diferențiabile. Dacă X și Y sunt două câmpuri de vectori tangenți hipersuprafeței, atunci aplicația $x \rightarrow g_x(X(x), Y(x))$ este diferențiabilă.
- ii) $(g_{ij}(x))$ este și matricea formei biliniare simetrice I_x relativă la baza canonica $\{e_1 = (1, 0, \dots, 0), \dots, e_n = (0, \dots, 0, 1)\}$ a spațiului vectorial $\mathbb{R}^n \cong T_x\mathbb{R}^n$.

Propoziție. Prima formă fundamentală a unei hipersuprafețe este pozitiv definită.

Propoziție (Invarianța primei forme fundamentale la o schimbare de parametri). Fie U și \bar{U} două mulțimi deschise în \mathbb{R}^n , $f : U \rightarrow E_{n+1}$ o hipersuprafață și $\varphi : \bar{U} \rightarrow U$ un difeomorfism. Atunci $\bar{f} = f \circ \varphi : \bar{U} \rightarrow E_{n+1}$ este hipersuprafață și avem $\bar{I}_{\bar{x}}(\bar{X}, \bar{Y}) = I_x(d\varphi_{\bar{x}}\bar{X}, d\varphi_{\bar{x}}\bar{Y})$, $\forall \bar{X}, \bar{Y} \in T_{\bar{x}}\mathbb{R}^n$, $\forall \bar{x} \in \bar{U}$, unde $x = \varphi(\bar{x})$.

Propoziție (Invarianța primei forme fundamentale la izometrii). Fie $f:U \rightarrow E_{n+1}$ o hipersuprafață și fie $B:E_{n+1} \rightarrow E_{n+1}$ o izometrie. Atunci:

- i) $\tilde{f} = B \circ f:U \rightarrow E_{n+1}$ este o hipersuprafață,
 - ii) pentru orice $x \in U$ și orice $X, Y \in T_{f(x)}f$ avem
- $$\tilde{g}_x(dB_{f(x)}X, dB_{f(x)}Y) = g_x(X, Y).$$

III.4. Forma a doua fundamentală a unei hipersuprafețe.

Invarianța formei a doua fundamentale la schimbări de parametri ce păstrează orientarea și la izometrii proprii. Hipersuprafețe ambilicale.
Aplicația Weingarten

Fie U o mulțime deschisă în \mathbb{R}^n și $f:U \rightarrow E_{n+1}$ o hipersuprafață. Considerăm reperul Gauss $\{f_{x^1}(x), \dots, f_{x^n}(x), N(x)\}$ într-un punct $f(x)$ al hipersuprafeței. Știm că imaginea aplicației Gauss $N:U \rightarrow E_{n+1} \cong T_{f(x)}E_{n+1}$ este inclusă în sfera unitate $S_n \subset E_{n+1}$.

Propoziție. Imaginea aplicației liniare

$$dN_x: \mathbb{R}^n \cong T_x\mathbb{R}^n \rightarrow \mathbb{R}^{n+1} \cong T_{f(x)}\mathbb{R}^{n+1}$$

este inclusă în $T_{f(x)}f$.

Propoziție. Aplicația $\Pi:T_x\mathbb{R}^n \times T_x\mathbb{R}^n \rightarrow \mathbb{R}$ definită prin

$$\Pi_x(X, Y) = -\langle dN_x X, df_x Y \rangle \quad (4.1.)$$

este o formă biliniară simetrică.

Definiție. Aplicația $x \rightarrow \Pi_x$ se numește **forma a doua fundamentală a hipersuprafeței**.

Observație. Bijecția liniară $df_x:\mathbb{R}^n \cong T_x\mathbb{R}^n \rightarrow T_{f(x)}f$ indică o formă biliniară simetrică $\Pi'_x:T_{f(x)}f \times T_{f(x)}f \rightarrow \mathbb{R}$ definită prin $\Pi'_x(X, Y) = \langle L_x X, Y \rangle$, unde am folosit notația

$$L_x = -dN_x \circ df_x^{-1}. \quad (4.2.)$$

În adevăr, pentru orice $X, Y \in T_{f(x)}f$ avem

$$\Pi'_x(X, Y) = \Pi_x(df_x^{-1}X, df_x^{-1}Y) = -\langle dN_x \circ df_x^{-1}X, Y \rangle = \langle L_x X, Y \rangle,$$

unde L_x este dat prin (4.2.). Se vede ușor că aplicația $L_x:T_{f(x)}f \rightarrow T_{f(x)}f$ este liniară. Aplicația liniară L_x se numește **aplicația lui Weingarten**. Aplicația $x \rightarrow \Pi'_x$ se numește tot **forma a doua fundamentală**.

Propoziție. Fie $(h_{ik}(x))_{1 \leq i, k \leq n}$ matricea formei biliniare simetrice H_x relativă la baza canonica $\{e_1, \dots, e_n\}$ a spațiului $\mathbb{R}^n \cong T_x \mathbb{R}^n$. Atunci $(h_{ik}(x))$ este și matricea formei biliniare simetrice H'_x relativă la baza canonica $\{f_{x^1}(x), \dots, f_{x^n}(x)\}$ a spațiului tangent $T_{f(x)}f$.

Propoziție.

- i) Funcțiile $h_{ik} : U \rightarrow \mathbb{R}$ sunt diferențiabile.
- ii) Dacă X și Y sunt două câmpuri de vectori tangenți hipersuprafeței, atunci funcția $x \rightarrow H_x(X(x), Y(x))$ este diferențiabilă.

Propoziție (Invarianța formei a două fundamentale la izometrii proprii).

Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață și $B : E_{n+1} \rightarrow E_{n+1}$, $Bv = Rv + v_0$ o izometrie proprie (adică matricea aplicației liniare R are determinantul egal cu unu). Atunci:

- i) $\tilde{f} = B \circ f : U \rightarrow E_{n+1}$ este o hipersuprafață,
 - ii) pentru orice $x \in U$ și orice $X, Y \in T_{f(x)}f$ avem
- $$\tilde{H}_x(dB_{f(x)}X, dB_{f(x)}Y) = H_x(X, Y).$$

III.5. Curburile principale ale unei hipersuprafețe. Curbura medie. Curbura totală (Gauss)

Considerăm o hipersuprafață $f : U \rightarrow E_{n+1}$ și fie $\{f_{x^1}(x), \dots, f_{x^n}(x), N(x)\}$ reperul Gauss într-un punct $f(x)$. Notăm cu $g_{ij}(x)$ coeficienții primei forme fundamentale a hipersuprafeței și cu $h_{ij}(x)$ coeficienții formei a două fundamentale, deci $g_{ij}(x) = \langle f_{x^i}(x), f_{x^j}(x) \rangle$, $h_{ij}(x) = \langle N(x), f_{x^i x^j}(x) \rangle$. Am văzut că aplicația Weingarten $L_x = -dN_x \circ df_x^{-1} : T_{f(x)}f \rightarrow T_{f(x)}f$ este liniară. Fie $(h_j^i(x))_{1 \leq i, j \leq n}$ matricea aplicației Weingarten, deci $L_x(f_{x^i}(x)) = h_i^k(x) f_{x^k}(x)$.

Propoziție.

- i) Operatorul liniar al lui Weingarten $L_x : T_{f(x)}f \rightarrow T_{f(x)}f$ este autoadjunct.
- ii) Rădăcinile ecuației $\det(h_j^i(x) - \rho(x) \delta_j^i) = 0$ sunt reale.

Definiție. Valorile proprii ale aplicației liniare Weingarten se numesc curburile principale ale hipersuprafeței.

Definiție. Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață și fie $\rho_1(x), \dots, \rho_n(x)$ curburile principale ale hipersuprafeței într-un punct oarecare $f(x)$. Un punct $f(x_0)$ al hipersuprafeței se numește

- i) **punct elliptic**, dacă $\rho_i(x_0) \neq 0$, $\forall i \in \{1, \dots, n\}$ și $\rho_1(x_0), \dots, \rho_n(x_0)$ au același semn,
- ii) **punct hiperbolic**, dacă $\rho_i(x_0) \neq 0$, $\forall i \in \{1, \dots, n\}$ și dacă există doi indici $i_0, j_0 \in \{1, \dots, n\}$ astfel încât $\rho_{i_0}(x_0)\rho_{j_0}(x_0) < 0$,
- iii) **punct parabolic**, dacă există doi indici $i_0, j_0 \in \{1, \dots, n\}$ astfel încât $\rho_{i_0}(x_0) = 0$, $\rho_{j_0}(x_0) \neq 0$,
- iv) **punct planar**, dacă $\rho_1(x_0) = \dots = \rho_n(x_0)$.

Definiție. Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață. Considerăm funcțiile $H, K : U \rightarrow \mathbb{R}$ definite prin

$$H(x) = \frac{1}{n}(\rho_1(x) + \dots + \rho_n(x)), \quad (5.1.)$$

$$K(x) = \rho_1(x) \dots \rho_n(x). \quad (5.2.)$$

unde $\rho_1(x), \dots, \rho_n(x)$ sunt curburile principale ale hipersuprafeței. $H(x)$ se numește **curbura medie**, iar $K(x)$ se numește **curbura totală sau curbura Gauss a hipersuprafeței** în punctul $f(x)$.

III.6. Simbolurile lui Christoffel ale unei hipersuprafețe. Formulele lui Gauss și Weingarten

Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață și fie $\{f_{x^1}(x), \dots, f_{x^n}(x), N(x)\}$ reperul Gauss într-un punct oarecare $f(x)$. Coeficienții primei forme fundamentale sunt date de formulele $g_{ij}(x) = \langle f_{x^i}(x), f_{x^j}(x) \rangle$. Considerăm funcțiile $|ij, k| : U \rightarrow \mathbb{R}$ și $|ij| : U \rightarrow \mathbb{R}$ definite prin

$$|ij, k| = \frac{1}{2} \left(\frac{\partial g_{jk}}{\partial x^i} + \frac{\partial g_{ik}}{\partial x^j} - \frac{\partial g_{ij}}{\partial x^k} \right), \quad (6.1.)$$

$$\left| \begin{matrix} k \\ ij \end{matrix} \right| = g^{ks} \left| \begin{matrix} ij, s \end{matrix} \right|. \quad (6.2.)$$

Definiție. Funcțiile $\left| \begin{matrix} ij, k \end{matrix} \right|$ (respectiv $\left| \begin{matrix} k \\ ij \end{matrix} \right|$) se numesc simbolurile lui Christoffel de prima (respectiv a doua) speță ale hipersuprafeței.

Propoziție (legea de transformare a simbolurilor lui Christoffel). La o schimbare de parametri, simbolurile lui Christoffel de speță a doua ale unei hipersuprafețe se transformă după legea

$$\frac{\partial^2 x^k}{\partial \bar{x}^i \partial \bar{x}^j} = - \left| \begin{matrix} k \\ rs \end{matrix} \right| \frac{\partial x^r}{\partial \bar{x}^i} \cdot \frac{\partial x^s}{\partial \bar{x}^j} + \left| \begin{matrix} r \\ ij \end{matrix} \right| \frac{\partial x^k}{\partial \bar{x}^r}. \quad (6.3.)$$

Propoziție. Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață și fie $\{f_{x^1}(x), \dots, f_{x^n}(x), N(x)\}$ reperul Gauss într-un punct oarecare $f(x)$ al hipersuprafeței. Fie h_{ij} coeficienții formei a doua fundamentale și fie h_j^i elementele matricei aplicației liniare a lui Weingarten. Avem formulele

$$(F.G.) \quad f_{x^i x^k} = \left| \begin{matrix} s \\ ik \end{matrix} \right| f_{x^s} + N h_{ik}$$

și

$$(F.W.) \quad N_{x^i} = -h_i^k f_{x^k}.$$

Formulele (F.G) se numesc **formulele lui Gauss**, iar formulele (F.W) se numesc **formulele lui Weingarten**.

III.7. A treia formă fundamentală a unei hipersuprafețe.

Interpretarea geometrică a curburii totale a unei suprafețe.

Formula lui Beltrami-Enneper. Formula lui Enneper

Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață și $\{f_{x^1}(x), \dots, f_{x^n}(x), N(x)\}$ reperul Gauss într-un punct oarecare $f(x)$ al hipersuprafeței. Considerăm forma biliniară simetrică $III_x : T_x \mathbb{R}^n \times T_x \mathbb{R}^n \rightarrow \mathbb{R}$ definită prin

$$III_x(X, Y) = \langle dN_x X, dN_x Y \rangle. \quad (7.1.)$$

Definiție. Aplicația $x \rightarrow III_x$ se numește **forma a treia fundamentală** și se notează cu III .

Observație. Bijecția liniară $d f_x : \mathbb{R}^n \cong T_x \mathbb{R}^n \rightarrow T_{f(x)} f$ induce o formă biliniară simetrică $III'_x : T_{f(x)} f \times T_{f(x)} f \rightarrow \mathbb{R}$ definită prin

$$III'_x(X, Y) = \langle L_x X, L_x Y \rangle, \quad (7.2.)$$

unde $L_x = -dN_x \circ df_x^{-1} : T_{f(x)} f \rightarrow T_{f(x)} f$ este aplicația liniară a lui Weingarten. În adevăr, pentru orice $X, Y \in T_{f(x)} f$ avem $III'_x(X, Y) = III_x(df_x^{-1}X, df_x^{-1}Y)$.

Aplicația $x \rightarrow III'_x$ o numim tot **forma a treia fundamentală**. Pe viitor vom nota III_x în loc de III'_x .

Propoziție. Fie $f : U \rightarrow E_3$ o suprafață, $K(x)$ și $H(x)$ curbura Gauss și curbura medie într-un punct $f(x)$ al suprafeței. Dacă I_x , II_x și III_x sunt cele trei forme fundamentale ale suprafeței atunci are loc **formula Beltrami-Enneper**

$$III_x - 2H(x)II_x + K(x)I_x = 0. \quad (7.3.)$$

Propoziție. Fie $f : U \rightarrow E_3$ o suprafață. Presupunem că, curbura totală $K(x)$ este negativă, oricare ar fi $x \in U$. Fie $x = f \circ c : I \rightarrow E_3$ o curbă în poziție generală pe suprafața f . Dacă c este linie asimptotică a suprafeței, atunci avem **formula Enneper**

$$K_2^2(t) = -K(x(t)), \quad \forall t \in I, \quad (7.4.)$$

unde $K_2(t)$ este torsiunea curbei c în punctul $c(t)$.

III.8. Simbolurile lui Riemann. Ecuațiile lui Gauss și Codazzi-Mainardi. Teorema Egregium (Gauss)

Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață. Notăm cu g_{ij} coeficienții primei forme fundamentale și cu $\begin{vmatrix} k \\ ij \end{vmatrix}$ simbolurile lui Christoffel de speță a doua. Considerăm funcțiile $R_{jkl}^i : U \rightarrow \mathbb{R}$, $R_{ijkl} : U \rightarrow \mathbb{R}$ definite prin

$$R_{jkl}^i = \frac{\partial \begin{vmatrix} i \\ jl \end{vmatrix}}{\partial x^k} - \frac{\partial \begin{vmatrix} i \\ jk \end{vmatrix}}{\partial x^l} + \begin{vmatrix} i \\ sk \end{vmatrix} \cdot \begin{vmatrix} s \\ jl \end{vmatrix} - \begin{vmatrix} i \\ sl \end{vmatrix} \cdot \begin{vmatrix} s \\ jk \end{vmatrix}, \quad (8.1.)$$

$$R_{ijkl} = g_{is} R_{jkl}^s. \quad (8.2.)$$

Definiție. Funcțiile R_{ijkl} (respectiv R_{jkl}^i) se numesc **simbolurile lui Riemann de prima** (respectiv **de a doua**) speță ale hipersuprafeței.

Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață și $\varphi : \bar{U} \rightarrow U$ o schimbare de parametri.

Vom nota cu g_{ij} , $\begin{vmatrix} k \\ ij \end{vmatrix}$, R_{ijkl} și R_{jkl}^i , (respectiv \bar{g}_{ij} , $\begin{vmatrix} k \\ ij \end{vmatrix}$, \bar{R}_{ijkl} și \bar{R}_{jkl}^i) coeficienții

primei forme fundamentale, simbolurile lui Christoffel de speță a doua, simbolurile lui Riemann de prima și a doua speță ale hipersuprafeței f (respectiv $\bar{f} = f \circ \varphi$).

Propoziție (legea de transformare a simbolurilor lui Riemann). *La o schimbare de parametri, simbolurile lui Riemann de speță a doua ale unei hipersuprafețe se transformă după legea*

$$\bar{R}_{\beta\gamma\delta}^{\alpha} \frac{\partial x^i}{\partial \bar{x}^\alpha} = R_{jkl}^i \frac{\partial x^j}{\partial \bar{x}^\beta} \cdot \frac{\partial x^k}{\partial \bar{x}^\gamma} \cdot \frac{\partial x^l}{\partial \bar{x}^\delta}, \quad (8.3.)$$

iar cele de prima speță se transformă după legea

$$\bar{R}_{\alpha\beta\gamma\delta} = R_{ijkl} \frac{\partial x^i}{\partial \bar{x}^\alpha} \cdot \frac{\partial x^j}{\partial \bar{x}^\beta} \cdot \frac{\partial x^k}{\partial \bar{x}^\gamma} \cdot \frac{\partial x^l}{\partial \bar{x}^\delta}. \quad (8.4.)$$

Propoziție.

i) a) $f_{x^i x^j x^k} = f_{x^i x^k x^j}$ dacă și numai dacă au loc ecuațiile Gauss

$$(E.G.) \quad R_{ijkl} = h_{ik} h_{jl} - h_{il} h_{jk}$$

și ecuațiile Codazzi-Mainardi

$$(E.C.) \quad \frac{\partial h_{ij}}{\partial x^k} + \left| \begin{matrix} s \\ ij \end{matrix} \right| h_{sk} = \frac{\partial h_{ik}}{\partial x^j} + \left| \begin{matrix} s \\ ik \end{matrix} \right| h_{sj},$$

unde R_{ijkl} sunt simbolurile lui Roemann de prima speță ale hipersuprafeței f .

ii) $N_{x^i x^j} = N_{x^j x^i}$ dacă și numai dacă au loc ecuațiile Codazzi-Mainardi.

Propoziție. *Simbolurile lui Riemann de prima speță au următoarele proprietăți*

- i) $R_{ijkl} + R_{ijlk} = 0,$
- ii) $R_{ijkl} + R_{jikl} = 0,$
- iii) $R_{ijkl} - R_{klji} = 0,$
- iv) $R_{ijkl} + R_{iklj} + R_{iljk} = 0.$

III.9. Derivare covariantă. Transport paralel Levi-Civita. Geodezice

Fie $f : U \rightarrow E_{n+1}$ o hipersuprafață și $c = f \circ x : I \rightarrow E_{n+1}$ o curbă pe hipersuprafața f . Fie X un câmp de vectori tangenți hipersuprafeței f în punctele curbei c , adică $X : I \rightarrow E_{n+1}$ este o aplicație diferențiabilă astfel încât $X(t) \in T_{c(t)} f$, $\forall t \in I$. Prin derivarea obișnuită a acestui câmp de vectori, obținem un câmp de vectori $\frac{dX(t)}{dt}$ de-a lungul curbei $c(t) = f \circ x(t)$ care în

general, nu mai este tangent la hipersuprafață f . Pentru a obține un câmp de vectori tangenți, considerăm proiecția acestui câmp de vectori în fiecare punct al curbei pe hiperplanul tangent la hipersuprafață în acel punct.

Notăm $\frac{\nabla X(t)}{dt} = \text{pr}_t \circ \frac{dX(t)}{dt}$, unde $\text{pr}_t : T_{c(t)} \mathbb{R}^{n+1} \rightarrow T_{c(t)} f$ este proiecția ortogonală de-a lungul normalei $N \circ x(t)$.

Definiție. Câmpul de vectori tangenți $\frac{\nabla X(t)}{dt}$ se numește **derivata covariantă a câmpului $X(t)$** .

Propoziție. Fie $X : I \rightarrow E_{n+1}$ un câmp de vectori tangenți hipersuprafeței $f : U \rightarrow E_{n+1}$ în punctele curbei $c = f \circ x$, deci $X(t) = X^k(t) f_{x^k}(x(t)) \in T_{f(x(t))} f$, unde $X^k : I \rightarrow \mathbb{R}$ sunt funcții diferențiabile pentru orice $k \in \{1, \dots, n\}$. Derivata covariantă a câmpului X este dată de formula

$$\frac{\nabla X(t)}{dt} = \left(\dot{X}^k(t) + \sum_{ij}^k (x(t)) X^i(t) \dot{X}^j(t) \right) f_{x^k}(x(t)), \quad (9.1.)$$

unde \sum_{ij}^k sunt simbolurile lui Cristoffel de speță a doua ale hipersuprafeței.

Propoziție. Fie $X, Y : I \rightarrow E_{n+1}$ două câmpuri de vectori tangenți hipersuprafeței $f : U \rightarrow E_{n+1}$ în punctele curbei $c = f \circ x : I \rightarrow E_{n+1}$ și fie $x \rightarrow g_x$ prima formă fundamentală a hipersuprafeței. Atunci avem:

$$\frac{d}{dt} g_{x(t)}(X(t), Y(t)) = g_{x(t)} \left(\frac{\nabla X(t)}{dt}, Y(t) \right) + g_{x(t)} \left(X(t), \frac{\nabla Y(t)}{dt} \right). \quad (9.2.)$$

Definiție. Fie $X : I \rightarrow E_{n+1}$ un câmp de vectori tangenți hipersuprafeței $f : U \rightarrow E_{n+1}$ în punctele curbei $c = f \circ x : I \rightarrow E_{n+1}$. X se numește **câmp de vectori paraleli** de-a lungul curbei c dacă

$$\frac{\nabla X(t)}{dt} = 0. \quad (9.3.)$$

Se mai spune că vectorul $X(t)$ se **transportă prin paralelism Levi-Civita** curba $c(t) = f \circ x(t)$.

Propoziție. Fie $f:U \rightarrow E_{n+1}$ o hipersuprafață și fie $X,Y:I \rightarrow E_{n+1}$ două câmpuri de vectori paraleli de-a lungul curbei $c=f \circ x$. Atunci

$$g_{x(t)}(X(t),Y(t)) = \text{constant}, \quad \forall t \in I. \quad (9.4.)$$

Teorema (transportului paralel). Fie $f:U \rightarrow E_{n+1}$ o hipersuprafață și $c(t)=f \circ x(t)$, $t \in [t_0, t_1]$ o curbă pe f . Atunci:

- i) pentru orice $X_0 \in T_{c(t_0)}f$ există un singur câmp de vectori paraleli $X(t, X_0)$, $t_0 \leq t \leq t_1$ de-a lungul curbei c cu $X(t_0, X_0) = X_0$;
- ii) aplicația $\|_c : T_{c(t_0)}f \rightarrow T_{c(t_1)}f$ definită prin $\|_c(X_0) = X(t_1, X_0)$ este un izomorfism liniar și o izometrie în raport cu produsul scalar introdus de prima formă fundamentală.

Definiție. Fie $c=f \circ x:I \rightarrow E_{n+1}$ o curbă regulată pe hipersuprafața $f:U \rightarrow E_{n+1}$. Presupunem că c este canonically parametrizată. Curba c se numește **geodezică** a hipersuprafeței dacă câmpul vectorial tangent $\dot{c}(s)$ este paralel, deci

$$\text{dacă } \frac{\nabla \dot{c}(s)}{ds} = 0.$$

Observație. Deoarece avem $\dot{c}(s) = \dot{x}^i(s) f_{x^i}(x(s))$, ecuația precedentă se scrie

$$\ddot{x}^k(s) + \left| \begin{matrix} k \\ ij \end{matrix} \right| (x(s)) \dot{x}^i(x) \dot{x}^j(s) = 0. \quad (9.5.)$$

Ecuațiile (9.5) se numesc **ecuațiile diferențiale ale geodezelor hipersuprafeței**.

Propoziție. Fie $f:U \rightarrow E_{n+1}$ o hipersuprafață, $x_0 \in U$ și $X_0 \in T_{f(x_0)}f$.

Pentru ε suficient de mic, există o singură geodezică $c:(-\varepsilon, \varepsilon) \rightarrow \mathbb{R}^{n+1}$, $c(s) = f \circ x(s)$, $|s| < \varepsilon$ cu $x(0) = x_0$ și $\dot{c}(0) = X_0$.

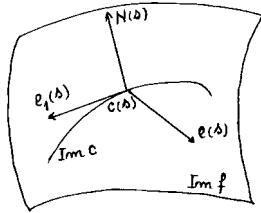
III.10. Suprafețe în spațiul euclidian tridimensional E_3

Reperul lui Darboux

Fie U o mulțime deschisă în E_2 și $x: I \subset \mathbb{R} \rightarrow U$ o curbă plană. Fie $f: U \rightarrow E_3$ o suprafață și $c = f \circ x: s \in I \subset \mathbb{R} \rightarrow c(s) \in E_3$, $\|c(s)\| = 1$, $\forall s \in I$ o curbă trasată pe suprafață. Într-un punct oarecare $c(s)$ al curbei considerăm trei vectori unitari și anume:

- vectorul unitar tangent $e_1(s) = \dot{c}(s)$,
- vectorul unitar $N(s) = N \circ x(s)$ normal la suprafață,
- vectorul unitar normal tangențial $e(s) = N(s) \times e_1(s)$.

Definiție. Reperul $\{e_1(s), e(s), N(s)\}$ se numește **reper Darboux**.



Formulele lui Darboux

Propoziție. Fie $f: U \rightarrow E_3$ o suprafață și fie $c = f \circ x: s \in I \rightarrow c(s) = f(x^1(s), x^2(s)) \in E_3$ o curbă parametrizată canonic trasată pe suprafață f . Fie $\{e_1(s), e(s), N(s)\}$ reperul Darboux într-un punct oarecare $c(s)$ al curbei. Atunci avem formulele lui Darboux

$$\dot{e}_1(s) = K_g(s)e(s) + K_N(s)N(s), \quad (10.1.)$$

$$\dot{e}(s) = -K_g(s)e_1(s) + T_g(s)N(s), \quad (10.2.)$$

$$\dot{N}(s) = -K_N(s)e_1(s) - T_g(s)e(s), \quad (10.3.)$$

unde:

$$K_g(s) = \left\langle e(s), \left(\ddot{x}^k(s) + \sum_{ij}^k (x(s)) \dot{x}^i(s) \dot{x}^j(s) \right) f_{x^k}(x(s)) \right\rangle, \quad (10.4.)$$

$$T_g(s) = \frac{1}{\sqrt{\det \|g_{ij}(x(s))\|}} \begin{vmatrix} g_{1i}(x(s))\dot{x}^i(s) & g_{2i}(x(s))\dot{x}^i(s) \\ h_{1k}(x(s))\dot{x}^k(s) & h_{2k}(x(s))\dot{x}^k(s) \end{vmatrix}, \quad (10.5.)$$

$$K_N(s) = h_{ij}(x(s))\dot{x}^i(s)\dot{x}^j(s). \quad (10.6.)$$

*Curbura geodezică. Torsiunea geodezică.
Curbura normală a unei curbe pe o suprafață*

Definiție. Considerăm funcțiile $K_g, T_g, K_N : I \rightarrow \mathbb{R}$ definite prin formulele (10.4.), (10.5.) și (10.6.). $K_g(s)$, $T_g(s)$ și $K_N(s)$ se numesc respectiv, **curbura geodezică, torsionea geodezică și curbura normală a curbei c în punctul $c(s)$.**

Linii de curbură

Definiție. Se numesc **linii de curbură** ale suprafeței curbele de pe suprafață cu proprietatea că în orice punct al lor torsionea geodezică este nulă.

Observație. Înănd seama de (10.5), obținem ecuația diferențială a liniilor de curbură

$$\begin{vmatrix} g_{1i}(x(s))\dot{x}^i(s) & g_{2i}(x(s))\dot{x}^i(s) \\ h_{1k}(x(s))\dot{x}^k(s) & h_{2k}(x(s))\dot{x}^k(s) \end{vmatrix} = 0. \quad (10.5').$$

Liniile de curbură ale unei suprafețe de rotație

Propoziție. Considerăm o suprafață de rotație

$$f : (x^1, x^2) \in U \rightarrow f(x^1, x^2) = (\varphi(x^1) \cos x^2, \varphi(x^1) \sin x^2, \psi(x^1)) \in E_3.$$

Dacă f nu are puncte ombilicale, atunci liniile de curbură sunt curbele coordonate ale suprafeței.

Caracterizări geometrice ale geodezicelor unei suprafețe

Propoziție. Fie $f : U \rightarrow E_3$ o suprafață și fie $c = f \circ x : I \rightarrow E_3$ o curbă trăsată pe suprafață f . Presupunem că c este în poziție generală. Următoarele afirmații sunt echivalente:

- (i) c este geodezică,
- (ii) Curbura geodezică în fiecare punct al curbei este nulă,
- (iii) Normala principală în orice punct al curbei c coincide cu normala la suprafață în acel punct,
- (iv) Planul osculator într-un punct oarecare al curbei este perpendicular pe planul tangent la suprafață în acel punct.

Propoziție. Fie $f : U \rightarrow E_3$ o suprafață și fie $c = f \circ x : s \in I \rightarrow c(s) \in E_3$, $\|\dot{c}(s)\| = 1$, $\forall s \in I$ o curbă pe f . Curba c este linie asimptotică a suprafeței dacă și numai dacă curbura normală a curbei c se anulează în fiecare punct al curbei.

Aplicații ale formulelor lui Darboux

Propoziție. Fie $f : U \rightarrow E_3$ o suprafață

- i) Dacă o geodezică este linie de curbură, atunci ea este curbă plană.
- ii) Dacă o geodezică este linie asimptotică, atunci ea este dreaptă.
- iii) Dacă o linie asimptotică este linie de curbură, atunci ea este curbă plană.

Definiție. Fie $f : U \rightarrow E_3$ o suprafață. Un vector nenul $X \in T_{f(x)}f$ se numește **vector principal al suprafeței în punctul $f(x)$** dacă reprezintă o valoare staționară a funcției $\rho : T_{f(x)}f \setminus \{0\} \rightarrow \mathbb{R}$, $\rho(Y) = \frac{II_x(Y, Y)}{I_x(Y, Y)}$. Numărul real $\rho(Y)$ se numește **curbura normală a suprafeței în punctul $f(x)$ după direcția vectorului Y** .

Teorema lui Rodriguez. Fie $f : U \rightarrow E_3$ o suprafață și fie $X \in T_{f(x)}f$, $X \neq 0$. Următoarele afirmații sunt echivalente:

- (i) X este un vector principal al suprafeței în punctul $f(x)$.
- (ii) X este vector propriu al aplicației liniare L_x a lui Weingarten.

III.11. Teorema fundamentală a teoriei hipersuprafețelor (Bonnet)

Am văzut că, fiind dată o hipersuprafață $f : U \rightarrow E_{n+1}$, putem asocia acestei hipersuprafețe funcțiile diferențiable $g_{ij} : U \rightarrow \mathbb{R}$ (g_{ij} sunt coeficienții primei formule fundamentale), $h_{ij} : U \rightarrow \mathbb{R}$ (h_{ij} sunt coeficienții celei de a doua formule fundamentale) și că aceste funcții verifică ecuațiile Gauss și ecuațiile Codazzi-Mainardi

Am văzut că ecuațiile lui Gauss și ecuațiile Codazzi-Mainardi sunt echivalente cu condițiile de integrabilitate pentru sistemul de ecuații

$$\begin{cases} f_{x^i x^j} = \begin{vmatrix} k \\ ij \end{vmatrix} f_{x^k} + h_{ij} N \\ N_{x^i} = -h_i^k f_{x^k} \end{cases}.$$

Apare naturală problema lui Bonnet:

Dându-se g_{ij} și h_{ij} putem să determinăm o hipersuprafață f care să admită pe g_{ij} și h_{ij} drept coeficienții primei forme fundamentale și respectiv ai celei de-a doua forme fundamentale? Dacă există o astfel de hipersuprafață, în ce condiții ea este unică? Răspunsul la această problemă este dat de teorema de mai jos.

Teorema lui Bonnet (teorema fundamentală a teoriei hipersuprafețelor).

Fie $U \subseteq \mathbb{R}^n$ o mulțime deschisă, stelată în raport cu originea. Presupunem date funcțiile diferențiabile $g_{ij} : U \rightarrow \mathbb{R}$, $h_{ij} : U \rightarrow \mathbb{R}$, $i, j \in \{1, \dots, n\}$ cu proprietățile $g_{ij} = g_{ji}$, $h_{ij} = h_{ji}$, matricea $(g_{ij})_{1 \leq i, j \leq n}$ este pozitiv definită, $\forall x \in U$. În plus, mai presupunem că g_{ij} și h_{ij} verifică ecuațiile Gauss și ecuațiile Codazzi-Mainardi, unde $\begin{vmatrix} k \\ ij \end{vmatrix}$ și R_{ijkl} sunt construite cu ajutorul lui g_{ij} . Atunci:

- i) există o hipersuprafață parametrizată $f : U \rightarrow E_{n+1}$, astfel încât g_{ij} sunt coeficienții primei forme fundamentale, iar h_{ij} sunt coeficienții celei de-a doua forme fundamentale.
- ii) două hipersuprafețe $f : U \rightarrow E_{n+1}$ și $\tilde{f} : U \rightarrow E_{n+1}$ care au pe g_{ij} (respectiv h_{ij}) drept coeficienți ai primei (respectiv ai celei de-a doua) forme fundamentale diferă printr-o izometrie proprie a spațiului euclidian E_{n+1} , adică există o izometrie proprie $B : E_{n+1} \rightarrow E_{n+1}$ astfel încât $\tilde{f} = B \circ f$.

BIBLIOGRAFIE

1. Nicolescu L., *Curs de geometrie*. Editura Fundației România de Mâine București, 2002 .
2. Ianuș S., *Curs de geometrie diferențială*. Tipografia Universității din București, 1981.
3. Teleman K., *Introducere în geometria diferențială*. Tipografia Universității București, 1986 .
4. Udriște C., *Curbe și suprafețe*. Tipografia Institutului Politehnic București, 1974.

LOGICĂ COMPUTAȚIONALĂ

Prof. univ. dr. GRIGORE ALBEANU

I. NOTIUNI INTRODUCTIVE

I.1. Gândirea

Gândire - proces psihic de reflectare a însușirilor esențiale și generale ale obiectelor și fenomenelor, a relațiilor dintre acestea, prin intermediul noțiunilor, judecăților și raționamentelor¹. [Definiție descriptiv-explicativă; **Zlate M.**, *Psihologia mecanismelor cognitive*, Ed. Polirom 1999; pag. 235]

Gândire - sistem ordonat de operații de prelucrare, interpretare și valorificare a informațiilor, bazat pe principiile abstractizării, generalizării și anticipării și subordonat sarcinii alegării alternativei optime din mulțimea celor posibile. [Definiție operațională; **Golu M. & Dicu A.**, *Introducere în psihologie*, Editura Științifică 1975, pag. 139]

Ipostaze ale gândirii [**Richard E. Mayer**, *Thinking, Problem Solving, Cognition*, W.H. Freeman & Co, New York, 1992] :

- "gândirea este cognitivă, dar este inferată din comportament, ea apare intern în minte sau în sistemul cognitiv, însă trebuie inferată indirect;
- gândirea este un proces care implică o manipulare sau un set de operații asupra cunoștințelor din sistemul cognitiv;
- gândirea este direcțională și rezultă în comportamentul care rezolvă o problemă sau este orientat către soluție".

Sunt considerate unelte ale gândirii [**Bernstein D.A., Roy E.J., Srull T.K. & Wickens C.D.** (1991), după **Zlate M.**, *op. cit.*]: concepte, propozițiile, silogismele, modelele mintale, scenariile, cuvintele, imaginile, algoritmii și euristicile.

Unitățile de bază ale gândirii [**Zlate M.**, *op. cit.*] sunt: *imaginea* (repräsentare mintală a unui obiect - unitate primitivă a gândirii); *simbolul* (unitate abstractă a gândirii ce redă obiectul, evenimentul, calitatea; cel mai simplu simbol este cuvântul); *conceptul* (eticheta pusă unei clase de obiecte); *prototipul* (exemplu ce ilustrează un concept); *operația* (acțiune reversibilă utilizată în formarea conceptelor sau la rezolvarea problemelor); *regula* sau *legea* (cea mai complexă unitate a gândirii; o relație între două sau mai multe concepte).

¹ Judecată (DEX-def_1): Facultatea de a gândi logic; rațiune, inteligență, gândire.

Judecată (DEX-def_2): Forma logică fundamentală exprimată printr-o propoziție în care se afirmă sau se neagă ceva.

Raționament (DEX-def_1): Înlănțuire logică de judecăți, care duce la o concluzie.

I.2. Limbajul

Limbajul poate fi privit ca funcție de utilizare a limbii în raporturile cu ceilalți oameni. Limba este un ansamblu de semne cu ajutorul cărora se poate comunica. Se distinge un vocabular, o mulțime de cuvinte și operații asupra cuvintelor.

Limba naturală (maternă sau nu) este utilă în experiența cotidiană. Limba artificială este creată de specialiști - limbajul artificial, pentru a se comunica, cu ajutorul unităților de bază ale gândirii, ceea ce se gândește, cum se gândește etc.

Unul sau mai multe cuvinte ale unei limbi [pentru comunicare] se constituie într-o propoziție. Propozițiile comunică întrebări, ordine, dorințe, dar și cunoștințe. Cele care comunică cunoștințe sunt numite *propoziții cognitive*.

Propozițiile cognitive pot fi *adevarate, false sau probabile!*

O propoziție cognitivă este adevărată numai dacă informația (cunoștința) pe care o exprimă corespunde stării de fapt despre care se vorbește [exemplu: Tabla are o formă dreptunghiulară].

Propoziția cognitivă este falsă dacă ceea ce ea susține nu corespunde unei stări de fapt [exemplu: Toate triunghiurile dreptunghice sunt isoscele].

Propoziția cognitivă este probabilă dacă nu poate fi stabilit nici adevărul și nici falsitatea propoziției [exemplu: Numărul numerelor întregi este impar].

În finalul secțiunii recomandăm reamintirea următoarelor principii ale logicii: identitate², noncontradicție³, terțul exclus⁴, rațiunea suficientă⁵, noțiuni⁶,

² *Principiul identității* : “Orice formă logică este ceea ce este.”; Orice propoziție p este echivalentă cu sine ($p \sim p$).

³ *Principiul noncontradicției*: “O propoziție nu poate fi, în același timp și sub același raport, atât adevărată, cât și falsă.” Exprimarea “în același timp” arată că anumite însușiri incompatibile pot reveni unui obiect, dar în momente diferite. Un om este Tânăr la o vîrstă și bătrân la o altă vîrstă. Exprimarea “sub același raport” spune că însușirile trebuie înțelese în același sens, pentru a nu se încalcă principiul identității. Același om poate fi “tânăr ca vîrstă”, dar “bătrân ca înfățișare”.

⁴ În același timp și sub același raport orice propoziție este fie acceptată, fie neacceptată, într-un sistem logic; a treia posibilitate este exclusă (latină: *tertium non datur*).

⁵ Orice propoziție are un temei. Prin “*rațiunea suficientă*” se înțelege “temei satisfacator”. Fiind dată o propoziție oarecare p, spunem că dispune de un temei satisfăcător Q numai dacă, dat fiind adevărul lui Q devine imposibil ca P să nu fie și el adevărat. Din adevăr trebuie să rezulte adevăr. Din fals poate rezulta orice (atât propoziții adevărate, cât și false).

⁶ *Noțiunea* este *forma logică* elementară care, în planul cunoașterii raționale, reprezintă reflectarea claselor de obiecte. *Forma lingvistică* ce materializează și comunică o noțiune are rol de “nume” pentru elementele clasei reflectate de noțiune. Ansamblul format dintr-o noțiune și un nume reprezintă un “termen”. Prin reflectarea clasei de obiecte, noțiunea reflectă ca un tot unitar, pe de o parte, însușirile (care formează “conținutul” sau *intensiunea noțiunii*) care conduc la delimitarea clasei, iar pe de altă parte, totalitatea obiectelor ce au aceste însușiri (care constituie “sfera” sau *extensiunea noțiunii*).

definiții⁷, clasificări⁸, propoziții categorice⁹, silogism¹⁰, inferențe deductive, inferențe inductive.

Pentru studiul individual solicitat se poate utiliza orice manual de liceu. Exemplificăm prin: Petre Bieltz, Anghelina Istrat, *Logica* - clasa a X-a, EDP 1990.

II. INTRODUCERE ÎN LOGICA SIMBOLICĂ

Descrierea unui proces algoritmic presupune, în multe momente, precizarea unor expresii logice care să direcționeze procesul de prelucrare. Aceste expresii trebuie să precizeze corect condițiile care trebuie îndeplinite, dar să fie simplu de evaluat. Prin urmare, cunoașterea bazelor *logicii matematice* și a regulilor de calcul utile în simplificarea expresiilor logice este obligatorie pentru orice programator¹¹. Materialul prezentat în continuare este preluat din [1]. Dreptul de preluare și valorificare aparține autorului.

Enunțuri și propoziții

Elementul principal al logicii matematice este *enunțul*. Enunțul este un *asamblaj* de semne susceptibil de a purta informații, în care putem distinge două părți fundamentale cu funcții specifice: subiectul (subiectele) și partea predicativă. Subiectul (subiectele) enunțului reprezintă entitatea (entitățile) din enunț despre care se comunică ceva. Ceea ce rămâne după eliminarea tuturor subiectelor este partea predicativă. Partea predicativă este unică, dar un enunț poate avea unul sau mai multe subiecte. Subiectele unui enunț pot fi determinate sau nu. Dacă toate

⁷ Definiția este operația logică prin care se precizează conținutul sau sfera noțiunii, înțelesul sau aria de aplicabilitate a unui nume. O definiție are trei părți: definitul (latina: *definiendum*) – reprezintă noțiunea sau numele care formează obiectul definiției; definitorul (latina: *definiens*) – ceea ce se spune despre obiectul definit, și relația de defnire (notată prin := sau \equiv_{df}).

⁸ Clasificarea este operația logică prin care noțiuni mai puțin generale sunt grupate în baza anumitor însușiri (sau *note*) în noțiuni mai generale.

⁹ Propozițiile categorice, sunt acele propoziții care exprimă, sub numai una din laturile sale, un raport între numai două noțiuni absolute. O proprietate importantă a propozițiilor categorice o reprezintă valoarea de adevăr a acestora. Valori posibile: adevărul (latina: *verum*), falsul (lat. *falsum*), ? (pentru proprietăți probabile). Proprietățile categorice au în structura lor un subiect logic S și un predicat logic P = ceea ce se spune despre subiectul logic. Propozițiile categorice sunt affirmative (raport de concordanță între S și P) sau negative (raport de opozиie între S și P). Pot fi folosiți cuantori asupra lui S (de universalitate, de existențialitate).

¹⁰ Silogismul este tipul fundamental de inferență deductivă a trei propoziții categorice: din (p, q) și (q, r) deducem (p, r).

¹¹Nu este suficient să fie rescrise, în limbajul de programare, expresiile logice formulate conform logicii limbajului natural. Pe de altă parte, pentru a se elabora algoritmi specifici unui domeniu concret, capacitatea de a gândi logic este o necesitate.

subiectele unui enunț sunt determinate avem de-a face cu o propoziție, în caz contrar enunțul se numește predicat. Subiectele nedeterminate se numesc *variabile libere*.

Exemplu: Enunțul “Secvența 1, 3, 5, 10, 11, 17 este crescătoare” este o propoziție. Enunțul “ $(x-1)(x+3) > 0$ ” este un predicat cu variabila liberă x . Din alt punct de vedere se disting *enunțuri atomice* (sau simple) și *enunțuri compuse*.

Enunțurile compuse sunt acele enunțuri care se obțin din alte enunțuri folosind:

- *operatori logici*: \neg (“non”), \wedge (“și”, “and”), \vee (“sau”, “or”), \rightarrow (“dacă ... atunci ...”, “if ... then ...”), \leftrightarrow (“dacă și numai dacă”, “if and only if”);
- *cuantificatori*: \forall (“oricare (ar fi)”), \exists (“există (cel puțin)”);
- *simbolurile* (și) pentru a specifica modul de structurare a unui enunț compus, atunci când nu sunt stabilite alte reguli.

Toate celelalte enunțuri sunt considerate atomice sau simple.

Un enunț compus format numai cu propoziții (numite și constante propoziționale) dă naștere tot la o propoziție. Dacă un enunț conține variabile propoziționale (simboluri care pot fi substituite cu constante propoziționale) atunci enunțul se mai numește și *formă propozițională*. De fapt o formă propozițională este o un predicat cu variabile libere propoziții.

În cele ce urmează vom nota cu A, B, \dots constantele propoziționale, iar cu p, q, r, \dots variabilele propoziționale. Prin P, Q, R, \dots vom nota enunțurile, indiferent de natura lor.

Definiție. Fie P un enunț. Enunțul $\neg P$ sau $\neg(P)$ se numește **negația** enunțului P . În limbaj natural, noul enunț se obține din P prin intercalarea cuvântului “nu” în fața părții predicative.

Definiție. Fie P și Q enunțuri. Enunțul $P \wedge Q$ sau $(P) \wedge (Q)$ se numește **conjuncția** enunțurilor P și Q . În limbaj natural, noul enunț se obține prin intercalarea cuvântului “și” între textele celor două enunțuri.

Definiție. Fie P și Q enunțuri. Enunțul $P \vee Q$ sau $(P) \vee (Q)$ se numește **disjuncția** enunțurilor P și Q . În limbaj natural, noul enunț se obține prin intercalarea cuvântului “sau” între textele celor două enunțuri.

Definiție. **Implicația** enunțurilor P și Q este enunțul $P \rightarrow Q$ sau $(P) \rightarrow (Q)$ care în limbaj natural se descrie prin “Dacă P atunci Q ”.

Definiție. Fie P și Q două enunțuri, enunțul $P \leftrightarrow Q$ sau $(P) \leftrightarrow (Q)$ se numește **echivalența** enunțurilor P și Q . În limbaj natural noul enunț se exprimă prin formularea: P dacă și numai dacă Q .

Definiție. Fie F un enunț în care variabila x este **liberă** (se mai folosește notația $F(x)$ sau Fx , F numindu-se variabilă *predicativă*, iar x variabilă *individuală*).

*Enunțul $\forall x F(x)$ se numește **cuantificarea universală** a enunțului F în raport cu variabila x . Dacă x este unica variabilă liberă a enunțului F atunci enunțul $\forall x F(x)$ este o propoziție.*

*Enunțul $\exists x F(x)$ se numește **cuantificarea existențială** a enunțului F în raport cu variabila x . Dacă x este unica variabilă liberă a enunțului F atunci enunțul $\exists x F(x)$ este o propoziție.*

*În enunțurile $\forall x F(x)$ și $\exists x F(\)$ variabila x se numește **variabilă legată**.*

Observație: Prin orice cuantificare numărul variabilelor libere ale unui predicat scade cu o unitate.

Limbajul logicii

Mulțimea constantelor propoziționale, a variabilelor propoziționale, a operatorilor logici, a simbolurilor (și) precum și a regulilor de formare date prin definițiile 1 - 5 care pornesc de la propoziții și formează alte propoziții, formează *limbajul logicii propoziționale*. Limbajul logicii predicatelor se obține integrând lista de simboluri și scheme propoziționale de mai jos împreună cu regulile de formare a expresiilor logice în *limbajul logicii predicatelor*.

Lista de simboluri și scheme propoziționale:

1. x, y, z, \dots variabile individuale; $a, b, c, \dots, x_1, y_1, x_2, y_2, \dots$, constante individuale;
2. F, G, H, \dots variabile predicative;
3. Fx, Fy, \dots scheme de funcții propoziționale. Funcția propozițională nu este încă propoziție, dar ea poate fi transformată în propoziție, fie prin substituirea variabilei cu o constantă, fie prin cuantificare.
4. $F(x, y), G(x, y, z), H(x, y, z, t), \dots$ predicate binare, ternare, și în general n -are;
5. Fa, Fb, Fc, \dots scheme de propoziții individuale;
6. $\forall x Fx, \forall y Gy, \dots$ scheme de propoziții universale;
7. $\exists x Fx, \exists y Gy, \dots$ scheme de propoziții existențiale;

Reguli de formare a expresiilor logice (formule bine formate) în cadrul teoriei limbajului logicii predicatelor:

1. Orice variabilă propozițională este expresie logică.
2. Orice schemă de funcție propozițională este expresie logică.
3. Dacă E este expresie logică atunci $\neg E$ este expresie logică.
4. Dacă $E(x)$ este o expresie logică cu x variabilă liberă atunci $\forall x E(x)$ și $\exists x E(x)$ sunt expresii logice.

5. Dacă E_1 și E_2 sunt expresii cu proprietatea că în ele una și aceeași variabilă nu apare într-o liberă și în cealălaltă cuantificată, atunci $E_1 \wedge E_2$, $E_1 \vee E_2$, $E_1 \rightarrow E_2$ și $E_1 \leftrightarrow E_2$ sunt expresii logice.

Valori de adevăr

Valoarea de adevăr a unei informații reprezintă gradul de concordanță dintre informația respectivă și contextul în care aceasta este enunțată. Din punctul de vedere al logicii tradiționale, sunt considerate numai enunțuri bivalente, adică enunțuri care pot avea doar două valori de adevăr: fals - simbolizat prin 0 - sau adevărat - simbolizat prin 1.

Orice propoziție este în mod obligatoriu fie adevărată, fie falsă. Nici o propoziție nu poate fi simultan și adevărată și falsă.

Referitor la predicate, putem întâlni situația în care un predicat este ambivalent. Mai precis, un predicat n -ar $F(x_1, x_2, \dots, x_n)$ este adevărat, respectiv fals, dacă toate propozițiile obținute din acest predicat, prin precizarea variabilelor sale în toate modurile posibile sunt adevărate, respectiv false. Deci stabilirea valorii de adevăr a unui predicat se reduce la stabilirea valorii de adevăr pentru toate propozițiile care generează predicatul.

Fie p și q propoziții oarecare. Valoarea de adevăr a propozițiilor compuse $\neg p$, $p \wedge q$, $p \vee q$, $p \rightarrow q$, $p \leftrightarrow q$ se poate determina pornind de la valorile de adevăr ale enunțurilor componente și semnificația operațiilor logice respective, conform următoarelor reguli:

Regula 1: Propoziția $\neg p$ este adevărată dacă și numai dacă propoziția p este falsă; rezultă că propoziția $\neg p$ este falsă dacă și numai dacă propoziția p este adevărată.

Regula 2: Propoziția $p \wedge q$ este adevărată dacă și numai dacă ambele propoziții p și q sunt adevărate; rezultă că propoziția $p \wedge q$ este falsă dacă cel puțin una dintre propozițiile p și q este falsă.

Regula 3: Propoziția $p \vee q$ este adevărată dacă și numai dacă cel puțin una dintre propozițiile p și q este adevărată; rezultă că propoziția $p \vee q$ este falsă dacă și numai dacă ambele propoziții p și q sunt false.

Regula 4: Propoziția $p \rightarrow q$ este o scriere prescurtată a enunțului $(\neg p) \vee q$.

Regula 5: Propoziția $p \leftrightarrow q$ este o scriere prescurtată a enunțului $(p \rightarrow q) \wedge (q \rightarrow p)$.

Propozițiile compuse care sunt adevărate independent de valorile de adevăr ale enunțurilor componente se numesc *legi logice* sau *tautologii*. Propozițiile care sunt false independent de valorile de adevăr ale componentelor se numesc

contradicții. Propozițiile care pot fi adevărate sau false în funcție de valorile particulare ale enușurilor componente se numesc propoziții *realizabile*.

Problema deciziei

Problema fundamentală a logicii matematizate este problema deciziei: Fiind dată o expresie logică să se decidă dacă ea reprezintă o lege logică (tautologie), o contradicție sau o funcție simplu realizabilă.

Această problemă poate fi soluționată prin mai multe mijloace dintre care amintim:

- a) prin eliminarea treptată a necunoscutelor;
- b) prin algoritmi;
- c) pe baza metodei axiomatice.

Eliminarea treptată a necunoscutelor se face pe baza unor raționamente prescurtate. Câteva reguli utile, care decurg din definiția valorii de adevăr a propozițiilor compuse, sunt:

- dacă un membru al conjuncției este fals, atunci conjuncția este falsă;
- dacă un membru al conjuncției este adevărat, atunci valoarea ei de adevăr depinde de restul componentelor;
- dacă un membru al disjuncției este adevărat, atunci disjuncția este adevărată;
- dacă un membru al disjuncției este fals, atunci valoarea ei de adevăr depinde de restul componentelor;
- dacă q este propoziție adevărată, atunci enunțul $p \rightarrow q$ este adevărat oricare ar fi valoarea de adevăr a propoziției p ;
- dacă p este propoziție falsă, atunci enunțul $p \rightarrow q$ este adevărat oricare ar fi valoarea de adevăr a propoziției q .

Metoda tablourilor de adevăr este utilă, dar dacă numărul componentelor este mare devine neficientă. Pentru o propoziție compusă care pornește de la n enușuri atomice sunt necesare 2^n rânduri în tabloul de adevăr, iar numărul de coloane depinde de structura propoziției. Pentru a avea cât mai puține coloane este necesară transformarea propoziției într-o echivalentă, dar cu număr minim de conjuncții.

O altă metodă elementară se bazează pe utilizarea regulilor de mai sus și scrierea propoziției supuse deciziei sub formă normală.

Lista principalelor legi logice (calculul propozițiilor)

Primul grup de tautologii exprimă proprietățile funcțiilor de adevăr denumite comutativitate, asociativitate, distributivitate, idempotență, reflexivitate și tranzitivitate. *Prin notația $E_1 = E_2$ înțelegem că formulele E_1 și E_2 sunt logic echivalente*, adică formula $E_1 \leftrightarrow E_2$ este identic adevărată (tautologie). Următoarele grupuri prezintă tautologiile clasice.

Conjuncția și disjuncția sunt comutative:

(1) $p \wedge q = q \wedge p$;

(2) $p \vee q = q \vee p$;

Conjuncția și disjuncția sunt asociative:

(3) $p \wedge (q \wedge r) = (p \wedge q) \wedge r$;

(4) $p \vee (q \vee r) = (p \vee q) \vee r$;

Conjuncția și disjuncția sunt distributive una față de alta:

(5) $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$;

(6) $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$;

Conjuncția și disjuncția sunt idempotente:

(7) $p \wedge p = p$;

(8) $p \vee p = p$;

Implicația și echivalența sunt reflexive și tranzitive:

(9) $p \rightarrow p$;

(10) $p \leftrightarrow p$;

(11) $((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$;

(12) $((p \leftrightarrow q) \wedge (q \leftrightarrow r)) \rightarrow (p \leftrightarrow r)$;

Relații de echivalență între expresiile logice:

În acest grup includem și operațiile de incompatibilitate (pentru “ p este incompatibil cu q ” scriem p / q), excludere (pentru “ p exclude q ” scriem $p W q$; operația sau exclusiv) și antidisjuncție (“nici p , nici q ”, pentru care scriem $p \downarrow q$).

(13) $(p \rightarrow q) = (\neg p) \vee q$;

(14) $p / q = (\neg p) \vee (\neg q)$; (definiția operatorului lui Sheffer)

(15) $p W q = ((\neg p) \wedge q) \vee (p \wedge (\neg q))$;

(16) $p \leftrightarrow q = ((\neg p) \vee q) \wedge ((\neg q) \vee p)$;

(17) $p \downarrow q = (\neg p) \wedge (\neg q)$;

(18) $\neg p = p / p$;

(19) $p \wedge q = (p / q) / (q / p)$;

(20) $p W q = (p / (q / q)) / (q / (p / p))$;

(21) $p \leftrightarrow q = ((p / (q / q)) / (q / (p / p))) / ((p / (q / q)) / (q / (p / p)))$;

(22) $p \downarrow q = ((p / p) / (p / q)) / ((p / p) / (q / q))$;

(23) $p \vee q = (p / p) / (q / q)$;

(24) $p \rightarrow q = p / (q / q)$;

- Legile absorbtiei:*
- (25) $p \wedge (p \vee q) = p$;
 - (26) $p \vee (p \wedge q) = p$;
- Legile dublei negatii:*
- (27) $(\neg(\neg p)) \rightarrow p$;
 - (28) $(\neg(\neg p)) = p$;
- Legile lui De Morgan:*
- (29) $\neg(p \wedge q) = (\neg p) \vee (\neg q)$;
 - (30) $\neg(p \vee q) = (\neg p) \wedge (\neg q)$;
- Legile implicației materiale:*
- (31) $p \rightarrow (q \rightarrow p)$; “Adevărul decurge din orice.”
 - (32) $(\neg p) \rightarrow (p \rightarrow q)$; “Falsul implică orice.”
- Legile reducerii la absurd:*
- (33) $(p \rightarrow (\neg p)) \rightarrow (\neg p)$;
 - (34) $((p \rightarrow q) \wedge (p \rightarrow (\neg q))) \rightarrow (\neg p)$;
- Conjuncția implică partea sa:*
- (35) $(p \wedge q) \rightarrow p$;
 - (36) $(p \wedge q) \rightarrow q$;
- Disjuncția este implicată de partea sa:*
- (37) $p \rightarrow (p \vee q)$
 - (38) $q \rightarrow (p \vee q)$;
- Legea “modus ponens”:* “Dacă este adevărat $p \rightarrow q$ și este adevărat p , atunci este adevărat q . ”
- (39) $((p \rightarrow q) \wedge p) \rightarrow q$;
- Legea “modus tollens”:*
- (40) $(p \vee q) \wedge (\neg q) \rightarrow p$;
- Legea noncontradicției:* “Nu este adevărat că p și $\neg p$. ”
- (41) $\neg(p \wedge (\neg p))$;
- Legea terțului exclus:* “Este adevărat p sau este adevărat $\neg p$. ”
- (42) $p \vee (\neg p)$;
- Legile excluderii:*
- (43) $(p \wedge q) \vee (p \wedge (\neg q)) = p$;
 - (44) $(p \vee q) \wedge (p \vee (\neg q)) = p$;
- Legea contrapozitiei:*
- (45) $(p \rightarrow q) \leftrightarrow ((\neg q) \rightarrow (\neg p))$;
- Legea silogismului:*
- (46) $(p \rightarrow q) \rightarrow ((q \rightarrow r) \rightarrow (p \rightarrow r))$.

Echivalențe logice în calculul predicatelor. Problema deciziei

Cu notațiile cunoscute au loc următoarele echivalențe logice:

- (1) $\forall x F(x) = \neg(\exists x \neg F(x));$
- (2) $\forall x \neg F(x) = \neg(\exists x F(x));$
- (3) $\exists x F(x) = \neg(\forall x \neg F(x));$
- (4) $\exists x \neg F(x) = \neg(\forall x F(x));$
- (5) $\forall x \forall y G(x, y) = \forall y \forall x G(x, y);$
- (6) $\exists x \exists y G(x, y) = \exists y \exists x G(x, y);$

Într-o formulă cuantificatorii ocupă diverse locuri. Pentru cuantificatori de același tip (omogeni) ordinea este indiferentă, nu același lucru se întâmplă pentru cuantificatori de tipuri diferite după cum rezultă din:

- (7) $\exists x \forall y G(x, y) \rightarrow \forall y \exists x G(x, y);$

În membrul stâng x este pentru toți y , iar în membrul drept x depinde de y . Rezultă că trecerea inversă nu este posibilă.

Proprietatea de distributivitate este întâlnită și la expresii ale calculului cu predicate:

- (8) $\forall x(F(x) \wedge G(x)) = (\forall x F(x)) \wedge (\forall x G(x));$
- (9) $\exists x(F(x) \vee G(x)) = (\exists x F(x)) \vee (\exists x G(x));$
- (10) $\exists x(F(x) \wedge G(x)) \rightarrow ((\exists x F(x)) \wedge (\exists x G(x)));$
- (11) $((\forall x F(x)) \vee (\forall x G(x))) \rightarrow \forall x(F(x) \vee G(x));$
- (12) $\forall x(F(x) \rightarrow G(x)) \rightarrow (\forall x F(x) \rightarrow \forall x G(x));$
- (13) $(\exists x F(x) \rightarrow \exists x G(x)) \rightarrow \exists x(F(x) \rightarrow G(x));$

Problema deciziei în logica predicatelor este mult mai complicată decât în calculul propozițiilor.

III. LATICI ȘI ALGEBRE BOOLEENE

Definiție. Se numește *latice* o mulțime nevidă L înzestrată cu două operații binare notate “ $+$ ” și “ \bullet ” astfel încât pentru oricare elemente $x, y, z \in L$ să fie valabile următoarele proprietăți:

comutativitate:

- (1) $x + y = y + x;$
- (2) $x \bullet y = y \bullet x;$

asociativitate:

- (3) $(x + y) + z = x + (y + z);$
- (4) $(x \bullet y) \bullet z = x \bullet (y \bullet z);$

absorbție:

$$(5) \quad x \bullet (x + y) = x;$$

$$(6) \quad x + (x \bullet y) = x.$$

Principiul dualității pentru latici: Dacă într-o propoziție adevărată din teoria laticilor se înlocuiește o operație prin cealaltă (și invers) se obține, de asemenea, o propoziție adevărată.

Propoziția 1. *In orice lattice $(L, +, \bullet)$, pentru orice element $x \in L$ sunt adevărate relațiile:*

$$(7) \quad x + x = x;$$

$$(8) \quad x \bullet x = x ..$$

Propoziția 2. *In orice lattice $(L, +, \bullet)$ avem $x + y = y$ dacă și numai dacă $x \bullet y = xx$, oricare ar fi elementele x și y aparținând mulțimii L .*

Definiție. Fie $x, y \in L$, notăm “ $x \leq y$ ” dacă și numai dacă $x + y = y$ (echivalent cu $x \bullet y = x$).

Propoziția 3. *Relația “ \leq ” este o relație de ordine.*

Observație: În cazul laticei $(P(M), \cup, \cap)$, relația de ordine corespunde relației de incluziune a două mulțimi.

Definiție. Fie $(L, +, \bullet)$ o lattice. Elementul $\pi \in L$ se numește **prim element** în L dacă pentru oricare (ar fi) $x \in L$ are loc relația $\pi \leq x$. Similar, elementul $\sigma \in L$ se numește **ultim element** în L dacă pentru oricare (ar fi) $x \in L$ are loc relația $x \leq \sigma$.

Propoziția 4. Fie latticea $(L, +, \bullet)$ cu $L = \{a_1, a_2, \dots, a_n\}$ formată cu n elemente (distincțe), atunci L are atât prim element cât și ultim element în raport cu relația de ordine definită mai sus și:

$$\pi = a_1 \bullet a_2 \bullet \dots \bullet a_n;$$

$$\sigma = a_1 + a_2 + \dots + a_n.$$

Definiție. Laticea $(L, +, \bullet)$ se numește **latice distributivă** dacă oricare ar fi elementele x, y, z din L sunt satisfăcute relațiile:

$$(9) \quad x + (y \bullet z) = (x + y) \bullet (x + z);$$

$$(10) \quad x \bullet (y + z) = (x \bullet y) + (x \bullet z).$$

Definiție. Laticea $(L, +, \bullet)$ se numește **latice complementată** dacă sunt îndeplinite condițiile:

(11) Există elementul neutru pentru operația “+” (numit element **nul**), notat cu 0, astfel încât $x + 0 = x$, oricare ar fi $x \in L$.

(12) Există elementul neutru pentru operația “•” (numit element **universal**), notat cu 1, astfel încât $x \bullet 1 = x$, oricare ar fi $x \in L$.

Pentru oricare $x \in L$ există $y_x \in L$, numit **complementul** lui x , care satisface relațiile:

$$(13) \quad x + y_x = 1;$$

$$(14) \quad x \bullet y_x = 0.$$

In cele ce urmează elementul y_x va fi notat prin x' .

Propoziția 5. Intr-o latice distributivă și complementată, complementul unui element este unic.

Definiție. O latice distributivă și complementată se numește **algebră booleană**.

În continuare vom considera că o algebră booleană sau, mai simplu, o algebră Boole este dată prin ansamblul $(B, +, \bullet, ', 0, 1)$, evidențiind mulțimea nevidă B , operațiile binare, operația unară (pentru obținerea complementului) și elementele neutre. De asemenea pentru $x \bullet y$ vom scrie, simplu, xy .

Se poate arăta că pentru a defini o algebră Boole este suficient ca, pentru oricare x, y, z elemente ale mulțimii B , să fie satisfăcute relațiile:

$$(1) \quad x + y = y + x;$$

$$(2) \quad xy = yx;$$

$$(3) \quad x + yz = (x + y)(x + z);$$

$$(4) \quad x(y + z) = xxy + xz;$$

$$(5) \quad x + yy' = x;$$

$$(6) \quad x(y + y') = x.$$

Atunci, pentru oricare x și y din B , avem $xx' = yy'$ (rezultat notat cu 0) și $x + x' = y + y'$ (rezultat notat cu 1).

Observații

1. Materialul din prezenta secțiune este preluat din [1]. Drepturile de preluare și valorificare aparține autorului.
2. În limbajul calculului propozițiilor, mulțimea claselor de echivalență (în sens algebric) a propozițiilor formează o algebră booleană dacă considerăm extinderea operațiilor de disjuncție, conjuncție și negație la nivelul claselor. Orice clasă conține propoziții echivalente logic, elementul nul este clasa contradicțiilor, iar elementul universal este clasa tautologilor.
3. *Mulțimea divizorilor unui număr M formează o algebră booleană față de operațiile definite astfel:*

$$a + b = \text{cmmmc}(a, b),$$

$$ab = \text{cmmdc}(a, b),$$

$$a' = M \text{ div } a \text{ (câtul împărțirii întregi a lui } M \text{ la numărul } a\text{)},$$

dacă și numai dacă descompunerea în factori primi a lui M nu conține decât factori la puterea 1 (adică M este liber de pătrate).

Exerciții

1. Folosind numai relațiile (1)-(6) să se demonstreze că, în orice algebră Boole $(B, +, \bullet, ', 0, 1)$, au loc relațiile (numite și reguli de calcul):
- (7) $x + x' = y + y'$ (notat prin 1), $xx' = yy'$ (notat prin 0), oricare $x, y \in B$.
 - (8) Pentru oricare $a \in B$ sistemul de ecuații $a + x = 1$, $ax = 0$ are soluția unică $x = a$.
 - (9) $x + 0 = x$, $x \bullet 1 = x$, oricare $x \in B$.
 - (10) $x + x = x$, $xx = x$, oricare $x \in B$.
 - (11) $x + 1 = 1$, $x \bullet 0 = 0$, oricare $x \in B$.
 - (12) $x + xy = x$, $x(x + y) = x$, oricare $x, y \in B$.
 - (13) $x + x'y = x + y$, $x(x' + y) = xy$, oricare $x, y \in B$.
 - (14) $(x + y) + z = x + (y + z)$, $(xy)z = x(yz)$, oricare $x, y, z \in B$.
 - (15) $(x + y)' = x'y'$, $(xy)' = x' + y'$, oricare $x + y \in B$.
 - (16) $(x')' = x$, oricare $x \in B$.
 - (17) $x + y = y$ dacă și numai dacă $xy = x$; $x, y \in B$.
 - (18) $x \leq x + y$, $y \leq x + y$, $xy \leq x$, $xy \leq y$; $x, y \in B$.
 - (19) Dacă $x \leq z$ și $y \leq z$ atunci $x + y \leq z$; $x, y, z \in B$.
 - (20) Dacă $t \leq x$ și $t \leq y$ atunci $t \leq xy$; $t, x, y \in B$.
 - (21) $0 \leq x \leq 1$, oricare $x \in B$.
 - (22) Dacă $x \leq y$, atunci $x + t \leq y + t$ și $xt \leq yt$, oricare $t, x, y \in B$.
 - (23) $x \leq y$ dacă și numai dacă $x' + y = 1$, dacă și numai dacă $xy' = 0$; $x, y \in B$.
 - (24) $x \leq y$ dacă și numai dacă $y' \leq x'$, $x, y \in B$.
 - (25) $x = y$ dacă și numai dacă $xy' + x'y = 0$; $x, y \in B$.
 - (26) $x = y$ dacă și numai dacă $(x + y')(x' + y) = 1$; $x, y \in B$.
 - (27) $x = y$ dacă și numai dacă $(x + y)(x' + y') = 0$; $x, y \in B$.
 - (28) $x = y$ dacă și numai dacă $xy + x'y' = 1$; $x, y \in B$.

IV. FUNCȚII BOOLEENE. FORME NORMALE

Fie $(B, +, \bullet, ', 0, 1)$ o algebră Boole, $n \geq 1$, număr natural. Notăm cu $F_n(B)$ mulțimea funcțiilor de n variabile definite pe B^n cu valori în B [1]. Este evidentă relația:

$$\text{card}(F_n(B)) = \text{card}(B)^{\text{card}(B)^n}.$$

Pentru $n = 2$ și $B = \{0, 1\}$, se obțin 16 funcții de 2 variabile, date prin tabelul:

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
0	1	0	0	0	0	1	1	1	0	0	0	0	1	1	1	1	
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	

sau prin formule (între “ “ trecem denumirea funcției conform logicii propozițiilor):

$f_0(x_1, x_2) = 0;$	“contradicția - funcția identic falsă“
$f_1(x_1, x_2) = x_1 x_2$	“conjuncția”
$f_2(x_1, x_2) = x_1 x'_2$	“negația implicației”
$f_3(x_1, x_2) = x_1$	“prima proiecție”
$f_4(x_1, x_2) = x'_1 x_2$	“negația implicației inverse”
$f_5(x_1, x_2) = x_2$	“a doua proiecție”
$f_6(x_1, x_2) = f_2(x_1, x_2) + f_4(x_1, x_2)$	“sau exclusiv”
$f_7(x_1, x_2) = x_1 + x_2$	“disjuncția, sau inclusiv”
$f_8(x_1, x_2) = x'_1 x'_2$	“ \downarrow , nici x_1 , nici x_2 ”
$f_9(x_1, x_2) = x_1 x_2 + x'_1 x'_2$	“echivalența”
$f_{10}(x_1, x_2) = x'_2$	“negația”
$f_{11}(x_1, x_2) = x_1 + x'_2$	“implicația inversă, $x_2 \rightarrow x_1$ ”
$f_{12}(x_1, x_2) = x'_1$	“negația”
$f_{13}(x_1, x_2) = x'_1 + x_2$	“implicația, $x_1 \rightarrow x_2$ ”
$f_{14}(x_1, x_2) = x'_1 + x'$	“/, x_1 este incompatibil cu x_2 ”
$f_{15}(x_1, x_2) = 1$	“lege logică sau tautologie”

Definiție. Numim **termen prim** atât o variabilă cât și complementul său.

Numim **conjuncție primă** orice termen prim și orice conjuncție de termeni primi.

Numim **disjuncție primă** orice termen prim și orice disjuncție de termeni primi.

Conjuncțiile de termeni primi se mai numesc și **monoame prime**. Monoamele prime (respectiv, disjuncțiile prime) în care apar toate variabilele sau complementul acestora, o singură dată, dar nu simultan variabila și complementul său, se numesc **monoame** (respectiv, **disjuncții**) **perfecte**.

Pentru n variabile, un monom perfect este de forma $x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}$, unde $i_k = 0, 1$ ($k = 1, 2, \dots, n$) cu convenția de notație $x_k^1 = x_k$, $x_k^0 = x'_k$. În general,

notăm, $x^1 = x$ și $x^0 = x'$ pentru oricare $x \in B$. Pentru $x = 3$, monoamele perfecte sunt: xyz , $x'yz$, $xy'z$, xyz' , $x'y'z$, $xy'z'$, $x'y'z'$. Analog se pot descrie disjuncțiile perfecte.

Definiție. Se numește **formă normală disjunctivă** (ND), disjuncția oricărui mulțimi de conjuncții prime. Se numește **formă normală conjunctivă** (NC), conjuncția oricărui mulțimi de disjuncții prime. Se numește **formă normală perfectă** acea formă normală formată numai cu monoame perfecte (în cazul ND) respectiv sume perfecte (în cazul NC).

Propoziția 1. Fie B algebra booleană de mai sus,

$$(c_{i_1, i_2, \dots, i_n})_{i_1, i_2, \dots, i_n \in \{0,1\}}, (d_{i_1, i_2, \dots, i_n})_{i_1, i_2, \dots, i_n \in \{0,1\}}$$

elemente din B și $x_1, x_2, \dots, x_n \in B$. Următoarele proprietăți sunt adevărate:

$$(1) \quad (i_1, i_2, \dots, i_n) \neq (j_1, j_2, \dots, j_n) \Rightarrow x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \bullet x_1^{j_1} x_2^{j_2} \dots x_n^{j_n} = 0,$$

($i_1, i_2, \dots, i_n, j_1, j_2, \dots, j_n \in \{0,1\}$),

$$(2) \quad \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} = 1,$$

$$\sum_{i_1, i_2, \dots, i_n \in \{0,1\}} c_{i_1 i_2 \dots i_n} x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} +$$

$$(3) \quad + \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} d_{i_1 i_2 \dots i_n} x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} = \\ = \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} (c_{i_1 i_2 \dots i_n} + d_{i_1 i_2 \dots i_n}) x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}$$

$$\left(\sum_{i_1, i_2, \dots, i_n \in \{0,1\}} c_{i_1 i_2 \dots i_n} x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \right) \bullet$$

$$(4) \quad \bullet \left(\sum_{i_1, i_2, \dots, i_n \in \{0,1\}} d_{i_1 i_2 \dots i_n} x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \right) =$$

$$= \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} (c_{i_1 i_2 \dots i_n} \bullet d_{i_1 i_2 \dots i_n}) x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}$$

$$\left(\sum_{i_1, i_2, \dots, i_n \in \{0,1\}} c_{i_1 i_2 \dots i_n} x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \right)' =$$

$$= \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} \left(c_{i_1 i_2 \dots i_n} \right)' x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}$$

Definiție. *Functiile booleene simple ale algebrei B sunt acele funcții booleene a căror expresie se obține plecând de la proiecții și aplicând operațiile algebrei booleene asupra unor elemente constituite anterior. Functiile booleene (la modul general) ale algebrei booleene se obțin ca și cele simple, dar luând elemente de plecare atât proiecțiile cât și constantele algebrei Boole B .*

Teorema 1. *O funcție $f : B^n \rightarrow B$ este booleană dacă și numai dacă există constantele $c_{i_1 i_2 \dots i_n} \in B$ ($i_1, i_2, \dots, i_n \in \{0,1\}$) astfel încât*

$$(6) \quad f(x_1, x_2, \dots, x_n) = \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} c_{i_1 i_2 \dots i_n} x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}, \quad \forall x_1, x_2, \dots, x_n \in B$$

Când (6) are loc, constantele $c_{i_1 i_2 \dots i_n}$ sunt unic determinate de

$$(7) \quad c_{i_1 i_2 \dots i_n} = f(i_1, i_2, \dots, i_n), \quad \forall i_1, i_2, \dots, i_n \in \{0,1\}.$$

Propoziția 2. *O funcție booleană de n variabile este simplă dacă și numai dacă*

$$(8) \quad f(i_1, i_2, \dots, i_n) \in \{0,1\}, \text{ pentru oricare } i_1, i_2, \dots, i_n \in \{0,1\}.$$

Propoziția 3. *O funcție booleană este funcție booleană simplă dacă și numai dacă satisface:*

$$(9) \quad f(x_1, \dots, x_n) = \sum_{\substack{i_1, i_2, \dots, i_n \in \{0,1\} \\ f(i_1, i_2, \dots, i_n) = 1}} x_1^{i_1} x_2^{i_2} \dots x_n^{i_n}, \quad \forall x_1, x_2, \dots, x_n \in B$$

Comentariu: Din cele de mai sus rezultă că mulțimea funcțiilor booleene pe B coincide cu mulțimea funcțiilor booleene simple (cu același număr de variabile) dacă și numai dacă $B = \{0,1\}$.

Teorema 2. *O funcție booleană f este simplă dacă și numai dacă se scrie astfel:*

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= \\ &= \prod f(i_1, i_2, \dots, i_n) + x_1^{i_1^0} + x_2^{i_2^0} \dots + x_n^{i_n^0}, \end{aligned}$$

luând în considerare notațiile și convenția de mai sus.

Comentariu. Teorema 2. pentru funcții booleene simple arată că orice funcție booleană simplă, în particular orice propoziție bine formată în contextul limbajului calculului propozițional, poate fi specificată în format NC, nu neapărat de tip perfect. De obicei se utilizează o expresie simplificată, echivalentă cu cea inițială, dar de tip conjunctiv. Să observăm, de asemenea, că funcțiile booleene simple pot descrie funcționarea oricărui sistem ale cărui componente pot avea cel mult două stări. Aceasta explică utilizarea rezultatelor din teoria funcțiilor booleene la studiul circuitelor sistemelor de calcul.

Simplificarea funcțiilor booleene simple

O problemă importantă legată de reprezentarea funcțiilor booleene o constituie reducerea la minimum a expresiei analitice a funcției booleene (mai puțini termeni și cât mai puțini factori). O funcție booleană poate avea mai multe forme disjunctive nesimplificabile.

O metodă pentru aflarea unei forme normale disjunctive minime constă în construirea tuturor formelor normale disjunctive ale funcției și alegerea uneia dintre cele mai scurte. Pentru un număr mare de variabile această metodă este neficientă.

Din punct de vedere algoritmic, problema găsirii unei forme minimale a unei funcții booleene este decidabilă. Algoritmul Quine-McCluskey (A se vedea Enescu Gh., Logica simbolică, Ed. Stiințifică, București, 1971) este destinat obținerii formei ND a unei funcții booleene prin utilizarea sistematică a regulilor de calcul într-o algebră Boole.

Exemplul 1 (Evaluarea expresiilor booleene): Fie B algebra Boole de mai sus și funcția f dată prin legea:

$$f(x_1, x_2, x_3) = x_1 x_2 + x_1 x'_2 x_3 + x'_1 x'_2 + x'_1 x_2 x'_3.$$

Forma ND perfectă este:

$$\begin{aligned} f(x_1, x_2, x_3) &= \\ &= x_1 x_2 x_3 + x_1 x_2 x'_3 + x_1 x'_2 x_3 + x'_1 x'_2 x_3 + x'_1 x'_2 x'_3 + x'_1 x_2 x'_3 \end{aligned}$$

și corespunde următorului tabel:

x	y	z	$f_{231}(x, y, z)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Forma NC perfectă este

$$f(x_1, x_2, x_3) = (x_1 + x'_2 + x'_3)(x'_1 + x_2 + x_3)$$

a cărei evaluare necesită 3 operații de negare (**not**), 4 disjuncții (**or**) și o conjuncție (**and**). După aplicarea regulilor de calcul asupra formei ND perfecte se pot obține următoarele exprimări posibile:

$$(*) \quad f(x_1, x_2, x_3) = x_1 x_2 + x'_2 x_3 + x'_1 x'_3 \text{ și}$$

$$(**) \quad f(x_1, x_2, x_3) = x_1 x_3 + x_2 x'_3 + x'_1 x'_3$$

ambele necesitând 3 operații **not**, 2 operații **or** și 3 operații **and**.

V. AXIOMATIZAREA LOGICII PROPOZIȚIILOR

V.1. Preliminarii

Un sistem axiomatic se construiește astfel:

- a) se specifică lista semnelor,
- b) se specifică regulile de formare a formulelor compuse din formule date,
- c) se specifică lista de axiome (și eventuale definiții),
- d) se specifică regulile de deducție. Pentru o altă abordare, complet formalizată, recomandăm consultarea lucrării [2].

Concepțele fundamentale ale unui sistem axiomatic sunt: definiția, regula și teorema. Definiția este o propoziție sau o formulă prin care unele expresii sunt introduse pe baza altora. Axioma este o propoziție (resp. o formulă) luată ca demonstrată în sistemul considerat. Regula este o propoziție cu ajutorul căreia din propoziții date (una sau mai multe), respectiv din formule date putem obține alte propoziții, respectiv formule. Teorema este orice propoziție (resp. formulă) dedusă din axiome pe baza regulilor de deducție.

V.2. Sistemul axiomatic Hilbert-Ackerman

Lista de simboluri:

Variabile propoziționale: p, q, p_1, p_2, \dots

și paranteze pentru a indica ordinea de efectuare a operațiilor.

Operatorii (de bază): \vee (or), \neg (not).

Se acceptă că \neg este mai prioritar decât \vee .

Operatori introdusi prin definiție: $\rightarrow, \wedge, \leftrightarrow, /, \downarrow$

Reguli de formare:

1. Variabilele propoziționale sunt formule;
2. Dacă A este formulă atunci $\neg A$ este de asemenea o formulă;
3. Dacă A și B sunt formule atunci $A \vee B, A \wedge B, A \rightarrow B, A \leftrightarrow B, A / B, A \downarrow B$ sunt de asemenea formule.

Definiții:

$$D_1 : p \rightarrow q = \neg p \vee q;$$

$$D_2 : p \wedge q = \neg(\neg p \vee \neg q);$$

$$D_3 : p \leftrightarrow q = \neg(\neg(p \vee q) \vee \neg(\neg q \vee p));$$

$$D_4 : p / q = \neg p \vee \neg q;$$

$$D_5 : p \downarrow q = \neg(p \vee q).$$

Axiome:

$$A_1 : (p \vee p) \rightarrow p;$$

$$A_2 : p \rightarrow (p \vee q);$$

$$A_3 : (p \vee q) \rightarrow (q \vee p);$$

$$A_4 : (p \rightarrow q) \rightarrow ((r \vee p) \rightarrow (r \vee q)).$$

Reguli de deducție:

Regula detașării (modus ponens): Dacă este demonstrat A și dacă este demonstrat $A \rightarrow B$ atunci este demonstrat B (separat, detașat de A). Scriere prescurtată: $\{A, A \rightarrow B \mid -B\}$.

Regula substituției: Într-o formulă A , o variabilă propozițională p poate fi substituită cu o formulă oarecare B , cu condiția ca variabila p să fie înlocuită cu formula B , oriunde apare în formula A . Notație: $s(A, p, B)$ sau $s(p, B)$ când formula A se subînțelege.

$$\text{Regula idempotentei disjuncției: } A \vee A \mid -A.$$

Considerăm axioma A_1 și efectuăm $s(p, A)$. Obținem: $(A \vee A) \rightarrow A$. Prin aplicarea regulii modus ponens obținem: $\{A \vee A, (A \vee A) \rightarrow A \mid -A\}$.

$$\text{Regula extinderii disjuncției: } A \mid -A \vee B.$$

Considerăm A_2 și efectuăm substituțiile $s(p, A)$, $s(q, B)$, adică: $A \rightarrow (A \vee B)$. Aplicăm regula modus ponens și obținem: $\{A, A \rightarrow (A \vee B) \mid -(A \vee B)\}$.

$$\text{Regula comutativității disjuncției: } (A \vee B) \mid -(B \vee A).$$

Considerăm A_3 și efectuăm substituțiile $s(p, A)$, $s(q, B)$, adică: $(A \vee B) \rightarrow (B \vee A)$. Aplicăm regula modus ponens și obținem: $\{A \vee B, (A \vee B) \rightarrow (B \vee A) \mid -(B \vee A)\}$.

$$\text{Regula extinderii disjunctive a termenilor implicatiei:}$$

$$(A \rightarrow B) \mid -((C \vee A) \rightarrow (C \vee B)).$$

Considerăm A_4 și efectuăm substituțiile: $s(p, A)$, $s(q, B)$ și $s(r, C)$, adică:

$$(A \rightarrow B) \rightarrow ((C \vee A) \rightarrow (C \vee B)).$$

Se aplică regula modus ponens și rezultă:

$$\{A \rightarrow B, (A \rightarrow B) \rightarrow ((C \vee A) \rightarrow (C \vee B)) \mid -(C \vee A) \rightarrow (C \vee B)\}.$$

Teoreme: Teorema este orice propoziție (resp. formulă) dedusă din axiome pe baza regulilor de deducție. Se poate introduce noțiunea de formulă demonstrabilă sau teoremă că fiind o formulă pentru care există o demonstrație formală, adică un sir de formule compus din substituții în *axiome sau formule demonstrabile* și formule obținute prin *reguli de deducție* din termeni anteriori ai sirului. Dacă α este formulă demonstrabilă numai din axiome scriem $\mid -\alpha$. Observăm că teoremele sunt formule demonstrabile din axiome. Dacă în afară de axiome, în deducerea formulei α , se adaugă și alte formule dintr-o mulțime H (pe care le vom numi

ipoteze) atunci spunem că α este deductibilă din familia de ipoteze H și scriem $H \vdash \alpha$.

Teorema 1. $\vdash (p \rightarrow q) \rightarrow ((r \rightarrow p) \rightarrow (r \rightarrow q))$.

Regula silogismului sau regula tranzitivității: Dacă este demonstrată formula $A \rightarrow B$ și este demonstrată formula $B \rightarrow C$ atunci rezultă că este demonstrată formula $A \rightarrow C$. Scriere prescurtată: $\{A \rightarrow B, B \rightarrow C\} \vdash (A \rightarrow C)$.

Teorema 2. $\vdash (\neg p \vee p)$.

Teorema 3. $\vdash (p \vee \neg p)$.

Teorema 4. $\vdash p \rightarrow (\neg \neg p)$.

Teorema 5. $\vdash (\neg \neg p) \rightarrow p$.

Teorema 6. $\vdash (p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$.

Regula substituirii expresiilor echivalente. Dacă două formule se deduc una din alta atunci pot fi puse una în locul alteia.

Exercițiu

Să se obțină demonstrații formale pentru următoarele afirmații::

1. $\vdash \neg(\neg(p \wedge q)) \rightarrow \neg p \vee \neg q$;
2. $\vdash \neg\neg p \vee \neg q \rightarrow \neg(p \wedge q)$;
3. $\vdash \neg(\neg(p \vee q)) \rightarrow \neg p \wedge \neg q$;
4. $\vdash \neg(\neg p \wedge \neg q) \rightarrow \neg(p \vee q)$;
5. $\vdash p \wedge q \rightarrow q \wedge p$;
6. $\vdash q \wedge p \rightarrow p \wedge q$;
7. $\vdash p \wedge q \rightarrow p$;
8. $\vdash p \wedge q \rightarrow q$;
9. $\vdash p \vee (q \vee r) \rightarrow q \vee (p \vee r)$;
10. $\vdash p \vee (q \vee r) \rightarrow (p \vee q) \vee r$;
11. $\vdash \neg(p \vee q) \vee r \rightarrow p \vee (q \vee r)$;
12. $\vdash p \wedge (q \wedge r) \rightarrow (p \wedge q) \wedge r$;
13. $\vdash \neg(p \wedge q) \wedge r \rightarrow p \wedge (q \wedge r)$;
14. $\vdash p \rightarrow (q \rightarrow p \wedge q)$
15. $\vdash p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$.

VI. PROPRIETĂȚILE SISTEMELOR AXIOMATICE

VI.1. Proprietățile sistemelor axiomatice

Un sistem axiomatic (SA) pentru a fi acceptat trebuie să satisfacă următoarele proprietăți formale:

Necontradicția – Un SA este necontradictoriu dacă și numai dacă, în el, nu poate fi demonstrată o formulă împreună cu negația ei.

Independenta – Un SA este independent dacă nici una dintre axioamele sale nu poate fi dedusă din restul axiomelor. Această cerință nu este socotită obligatorie de toți “logicienii”.

Completitudinea – Un SA este complet dacă adăugând la SA o formulă F nedemonstrabilă în SA, obținem o contradicție.

Aceste proprietăți se verifică, de obicei, cu ajutorul unei interpretări, adică se trece de la aspectul sintactic la cel semantic.

VI.2. Teorema de consistență

Propoziția 1.

- Pentru oricare $m > 0$, $A_1, A_2, \dots, A_m \vdash A_i$, $i = 1, 2, \dots, m$.
- Pentru oricare m și p numere naturale, dacă $A_1, A_2, \dots, A_m \vdash B_i$, $i = 1, 2, \dots, p$, iar $B_1, B_2, \dots, B_p \vdash C$, atunci $A_1, A_2, \dots, A_m \vdash C$.

Propoziția 2.

- Dacă $\vdash A \rightarrow B$, atunci $A \vdash B$.
- Pentru oricare $m > 0$, dacă $A_1, A_2, \dots, A_{m-1} \vdash A_m \rightarrow B$, atunci $A_1, \dots, A_m \vdash B$.

Corolar. Dacă $\vdash A_1 \rightarrow (A_2 \rightarrow (\dots (A_{m-1} \rightarrow (A_m \rightarrow B)) \dots))$, atunci $A_1, \dots, A_m \vdash B$.

Teorema 1 [Teorema deducției].

- Dacă $A \vdash B$, atunci $\vdash A \rightarrow B$;
- Dacă $A_1, A_2, \dots, A_{m-1}, A_m \vdash B$, atunci $A_1, A_2, \dots, A_{m-1} \vdash A_m \rightarrow B$.

Teorema 2 [de consistență pentru calculul propozițiilor]. Orice teoremă (formulă demonstrabilă din axiome) este o tautologie.

Corolar [Proprietatea necontradicției]. Nu există formulă B astfel încât $\vdash B$ și $\vdash \neg B$.

Propoziția 3. Au loc următoarele reguli de includere/eliminare.

- a) Dacă $H, A \vdash B$, atunci $H \vdash A \rightarrow B$;
- b) $A, B \vdash A \wedge B$;
- c) $A \vdash A \vee B$, $B \vdash A \vee B$;
- d) Dacă $H, A \vdash B$ și $H, A \vdash \neg B$, atunci $H \vdash \neg A$;
- e) Dacă $A \rightarrow B$ și $B \rightarrow A$, atunci $A \leftrightarrow B$;
- f) $A, A \rightarrow B \vdash B$;
- g) $A \wedge B \vdash A$, $A \wedge B \vdash B$;
- h) Dacă $H, A \vdash C$ și $H, B \vdash C$, atunci $H, A \vee B \vdash C$;
- i) $\neg \neg A \vdash A$, $A, \neg A \vdash B$;
- j) $A \leftrightarrow B \vdash A \rightarrow B$ și $A \leftrightarrow B \vdash B \rightarrow A$.

Propoziția 4. Fiecărei intrări (rând) din tabelul de adevăr al oricăruiu dintre operatorii considerați îi corespunde o deducție.

Propoziția 5. Fie E o formulă cu atomii P_1, P_2, \dots, P_n . Considerăm tabelul de adevăr pentru E . Atunci pentru fiecare din cele 2^n rânduri ale tabelului există o deducție.

Propoziția 6. Dacă formula E din prop. 6.5 este tautologie, atunci

$$P_1 \vee \neg P_1, \dots, P_n \vee \neg P_n \vdash E.$$

Teorema 3 [De completitudine – calculul propozițiilor]. Dacă E este tautologie atunci E este teoremă.

VII. CALCULUL AXIOMATIC AL PREDICATELOR

VII.1. Forme normale

Noțiunea de formă normală din calculul propozițiilor se poate generaliza și la calculul predicatelor.

Definiție. Dacă o formulă A din calculul predicatelor este transformată într-o formulă B astfel încât A și B au aceeași valoare de adevăr, în B nu apar alți operatori decât \wedge , \vee , \neg , \forall și \exists , iar negația (dacă există) se aplică numai la variabilele predicative și propoziționale, atunci B se numește redusa lui A .

Definiție [Formă normală prenex]. O formă redusă este **formă normală prenex** dacă nu conține operatorii \forall și \exists și, este formă normală booleană sau dacă operatorii de cuantificare formează un prefix și domeniul de acțiune al prefixului este în forma normală booleană.

Aducerea unei expresii la forma normală prenex se face pe baza următoarelor reguli de scoatere a cuantificatorilor în prefix:

1. Dacă expresia este fără cuantificatori se procedează exact ca în calculul propozițiilor;
2. Dacă expresia conține cuantificatori atunci se redenumesc variabilele astfel încât fiecare cuantificator să lege o variabilă diferită, se scot cuantificatorii în față, în ordinea în care apar în formulă, iar domeniul de acțiune este adus la una din formele normale booleene.

Regula de redenumire: O variabilă legată poate fi înlocuită cu o altă variabilă (care după înlocuire va apărea, de asemenea, legată) în întreg domeniul de acțiune din care ea face parte, cu condiția ca după înlocuire să se obțină din nou formula inițială.

Teorema 1. *Oricărei formule îi corespunde o formă normală echivalentă cu ea.*

Definiție [Formă normală Skolem]. *O formulă este formă normală Skolem dacă ea este o transformare a formelor normale prenex astfel încât nici un cuantificator existential (dacă există) nu urmează după vreun cuantificator universal. Relația dintre formula inițială și cea de tip Skolem este de echivalență deductivă: Dacă $M, A \vdash B$ și $M, B \vdash A$, atunci A este deductiv echivalentă cu B .*

VII.2. Sistemul axiomatic Hilbert-Ackerman

La axiomele calculului propozițional se adaugă și *axiomele*:

$$A_5 : \forall x Fx \rightarrow Fy ;$$

$$A_6 : Fy \rightarrow \exists x Fx .$$

Reguli de deducție [primare (1-4) și deriveate (5-6)]:

1. Regula modus ponens (formularea se păstrează) : PR_MP
2. Regulile substituției [PR1, PR2, PR3]
3. Regulile cuantificatorilor [PR4, PR5]
4. Regula redenumirii [PR6]
5. Regula universalității [PR7]
6. Regula substituției multiple [PR8]

Regulile substituției:

PR1 [*Substituția variabilelor propoziționale*] Într-o formulă A putem înlocui o variabilă propozițională cu o formulă B dacă se respectă condițiile:

- a) variabila propozițională este înlocuită pretutindeni unde apare în A ;
- b) B nu conține variabile individuale libere care sunt legate în A sau variabile individuale legate care în A sunt libere;
- c) dacă variabila propozițională se află în domeniul de acțiune al unui cuantificator atunci variabila legată de acest cuantificator nu se află în B .

PR2 [Substituția unei variabile individuale liberă] O variabilă individuală liberă poate fi înlocuită cu orice altă variabilă individuală liberă dacă:

- a) substituția se face oriunde apare variabila în formulă;
- b) variabila cu care o substituim nu apare legată în formula considerată.

PR3 [Substituția variabilelor predicative] Fie o formulă A care conține predicatul F , iar F conține n variabile individuale x_1, x_2, \dots, x_n , libere sau legate. F poate fi înlocuit cu o formulă B care conțin cel puțin n variabile libere dacă:

- a) variabilele libere ale lui B nu apar legate în A ;
- b) variabilele legate ale lui B nu apar libere în A ;
- c) În fiecare caz de apariție a lui F în A variabilele lui F sunt înlocuite numai cu asemenea variabile care nu apar legate în B .

Regulile cuantificatorilor

PR4. Din $A \rightarrow B(x)$ se deduce $A \rightarrow \forall x B(x)$, dacă x este variabilă liberă în B și nu apare în A .

PR5. Din $B(x) \rightarrow A$ se deduc $\exists x B(x) \rightarrow A$, dacă x este variabilă liberă în B și nu apare în A .

Regula redenumirii

PR6. O variabilă legată poate fi înlocuită cu o altă variabilă (ce după înlocuire va fi tot legată), în întreg domeniul de acțiune din care face parte, cu condiția ca după înlocuire să se obțină din nou formula și ca noua variabilă legată să nu apară liberă în formula inițială.

Regula universalității

PR7. Dacă formula $A(x)$ este demonstrabilă (x este variabilă liberă), atunci este demonstrabilă formula $\forall x A(x)$.

Regula substituției simultane

PR8. Fie formula $A(x_1, x_2, \dots, x_m, y_1, y_2, \dots, y_n)$ în care variabilele x_i sunt libere și variabilele y_j sunt legate (în totalitate). Variabilele x_i pot fi înlocuite cu alte variabile z_i , iar y_j cu variabile u_j cu condiția ca locurile în care stau variabile identice (în A) să fie puse variabile identice, iar locurile în care apar variabile diferite să fie puse variabile diferite.

Teoreme

1. $\vdash \forall x (Fx \vee \neg Fx)$; Terțul exclus generalizat.
2. $\vdash \forall x Fx \rightarrow \exists x Fx$.
3. $\vdash \forall x (A \vee Fx) \rightarrow A \vee \forall x Fx$.

Regula PR9. Dacă $\vdash A \rightarrow (B \rightarrow C(x))$, atunci $\vdash A \rightarrow (B \rightarrow \forall x C(x))$.

VII.3. Proprietățile sistemului axiomatic al predicatorilor

Pentru simplificare vom considera că se lucrează cu domenii finite.

- Necontradicția:* Se arată că axiomele sunt formule valide. Pentru $A_1 - A_4$, s-a demonstrat în cadrul calculului propozițional. Axiomele A_5 și A_6 prin eliminarea cuantificatorilor și variabilelor libere și tratarea variabilelor predicative ca și variabile propoziționale se transformă în teorema calculului propozițional $p \rightarrow P$. Apoi se arată că regulile de deducție transformate corespunzător duc de la formule cu valoarea 1 numai la formule cu valoarea 1.
- Completitudinea.* Calculul predicatorilor nu este complet în sensul completitudinii calculului propozițional. Există formula $\exists x Fx \rightarrow \forall x Fx$ (falsă când domeniul lui x este mai mare de un element) care adăugată la sistemul de axiome $A_1 - A_6$, nu-l transformă într-unul contradictoriu.

Gödel a demonstrat că fiecarei formule universal-valabile îi corespun o formă normală Skolem care este deductibilă din sistemul de axiome. Deci dacă orice formă normală Skolem universal valabilă se deduce din axiomele predicatorilor atunci sistemul este complet.

Teorema lui Gödel despre completitudine. *Orice formulă universal valabilă a logicii predicatorilor este demonstrabilă în limbajul calculului predicatorilor.*

VIII. LIMBAJUL PROLOG ȘI LOGICA PREDICATELOR

VIII.1. Introducere

PROgramarea LOGică (PROLOG) constă în transformarea logicii în calcul. Limbajul PROLOG are la bază teoria calculului cu predicate de ordinul întâi¹² și a fost dezvoltat în 1975, la Universitatea din Marsilia.

În logica calculului predicatorilor, obiectele lumii reale sunt modelate cu ajutorul simbolurilor. Aceștia pot fi: simboluri de constante, simboluri de variabile, simboluri pentru termeni compuși. Un simbol constant indică un singur individ sau concept. Un simbol de variabilă se referă, la momente diferite, la indivizi diferenți.

Termenii compuși se obțin cu ajutorul unui simbol funcțional (functor), aplicat asupra unor simboluri, numite argumente.

¹² Schemele care rezultă prin aplicarea variabilelor predicative asupra unor variabile individuale sunt scheme de ordinul întâi. De asemenea, aplicarea cuantificatorilor asupra variabilelor individuale conduce la scheme de ordinul întâi. Aplicarea cuantificatorilor asupra variabilelor predicative conduce la scheme de ordin doi. Pentru mai multe detalii se poate studia teoria tipurilor introdusă de Bertrand-Russel.

VIII.2. Forma clauzală

În PROLOG formulele calculului cu predicate trebuie scrise sub formă clauzală (normală). Aducerea unei formule date la forma normală impune parcurgerea a şase etape:

- eliminarea implicației [$p \rightarrow q = \neg p \vee q$];
- mutarea negației în interior (de exemplu $\text{not}(x)$ or $\text{not}(y)$ în loc de $\text{not}(x \text{ and } y)$). A se vedea și celealte reguli, inclusiv asupra cuantificatorilor $\neg(p \wedge q) = \neg p \vee \neg q$, $\neg(p \vee q) = \neg p \wedge \neg q$, $\neg(\exists x Fx) = \forall x (\neg Fx)$, $\neg(\forall x Fx) = \exists x (\neg Fx)$.
- Skolemizarea formulelor: O formulă de tipul $\exists x Fx \wedge G(x, y)$ este înlocuită cu $Fx_1 \wedge G(x_1, y)$. Dacă formula conține cuantificator universal, atunci e mai dificil. Formula “Orice om are o mama” $[\forall x \text{Om}(x) \rightarrow \exists y \text{Mama}(x, y)]$ poate fi prelucrată astfel: $[\forall x \text{Om}(x) \rightarrow \text{Mama}(x, G(x))]$, unde $G(x)$ furnizează $\text{Mama}(x)$.
- Scoaterea cuantificatorilor universali în exterior: La această fază cuantificatorii existențiali au fost deja eliminați prin procedeul de la etapa c. Formula $\forall x (Fx \rightarrow \forall y (G(y) \rightarrow Z(x, y)))$ este transformată în $\forall x \forall y (Fx \rightarrow (Gy \rightarrow Z(x, y)))$.
- Aplicarea legii distribuției lui *and* față de *or*: $[(A \wedge B) \vee C = (A \vee C) \wedge (B \vee C); A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)]$. De exemplu, $Fx \vee (G(y, x) \wedge (Py \vee Qy))$ se transformă în $(Fx \vee G(y, x)) \wedge (Fx \vee (Py \vee Qy))$.
- Scrierea în formă clauzală [Obținerea formei normale conjunctive care va furniza clauzele] Exemplificăm prin formula:

$$(\text{persoana}("Adam") \wedge \text{persoana}("Eva")) \wedge$$

$$\wedge ((\text{persoana}(X) \vee \neg \text{mama}(X, Y)) \vee \text{persoana}(Y))$$

care are trei cluze: Clauza 1: $\text{persoana}("Adam")$; Clauza 2: $\text{persoana}("Eva")$; Clauza 3, compusă din trei elemente: $\text{persoana}(X)$, $\neg \text{mama}(X, Y)$, $\neg \text{persoana}(Y)$.

```

predicates
    persoana(string)
    mama(string, string)
clauses
    persoana("Adam").
    persoana("Eva").
    persoana(X) :- mama(X, Y), persoana(Y).
  
```

Forma clauzală a formulei

$$(Fx \vee G(y, x)) \wedge (Fx \vee (Py \vee Qy)),$$

unde F = este _în_ vacanță, G = lucrează; P = furios, Q = trist și Y = “Andrei”, cuprinde două clauze, care nu au corespondent direct în PROLOG.

```
este_in_vacanta(X); lucreaza("Andrei", X) :- !.  
este_in_vacanta(X); furios("Andrei"); trist("Andrei") :- !.
```

IX. TIPURI DE DATE ȘI MECANISME PROLOG

IX.1. Structura programelor PROLOG

Programatorul PROLOG declară legile care guvernează universul problemei de rezolvat. Specifică datele și ce anume se cere.

Secțiunile PROLOG sunt: constants, domains, predicates, goal, clauses, etc.

Clauses conține fapte și reguli. Argumentele unei clauze sunt într-o ordine bine definită, fiecare argument având un anumit tip. Tipul argumentelor, numele predicatului care descrie un fapt sau o regulă și numărul argumentelor rezultă din declarațiile specificate în secțiunea **predicates**. Dacă mai multe predicate au același nume, dar aritate (număr de argumente) diferită, atunci predicatele sunt diferite. Identificarea va fi corectă datorită implementării mecanismului polimorfic.

Numele unui predicat este un sir (cu maxim 250) de litere, cifre și “_” care începe obligatoriu cu o literă mică. Argumentele pot fi: unul, mai multe sau nici unul. Un predicat este de forma:

```
nume_predicat(tip_arg1, tip_arg2, ..., tip_argn)
```

Faptele sunt de forma:

```
nume_predicat(arg1, arg2, ..., argn).
```

Regulile se pot descrie prin

```
nume_predicat(arg1, arg2, ..., argn) :- expresie_logica.
```

Argumentele pot fi și variabile. Ele apar, de obicei, în expresia logică din membrul drept.

O expresie logică utilizează conjuncții (specificate prin virgula (,) sau and), disjuncții (specificate prin punct_ști_virgulă (;) sau or), operatorul **not** aplicat către unui predicat (pentru a specifica negația), predicatul **fail** (pentru a introduce nedeterminismul) și predicatul **cut** (**tăietură**, specificat prin !), care are rolul de a opri căutarea prin *mecanismul backtracking*. Observați că, mai sus, au apărut conjuncțiile și tăietura.

Dacă goal nu apare în program, spunem că se lucrează în mod interactiv, prin interpretarea și identificarea entităților din program. La executare se activează fereastra Dialog și putem lansa întrebări.

Când secțiunea goal apare într-un program Turbo Prolog, ea trebuie să fie plasată după predicates, înainte de clauses. Ea este o clauză sau un sir de clauze conectate logic prin and (.) sau or (;). Se va termina cu punct. La executare se va obține o singură soluție (prima care s-a "potrivit"). Nu se face "backtracking".

Secțiunea **constants** permite declararea unor constante. Secțiunea poate apărea oriunde în program, cu condiția ca o constantă să fi fost declarată înainte de a fi utilizată.

Programele PROLOG pot conține comentarii incluse între /* și */.

De exemplu:

```
/* Aici sunt definite cateva constante */
```

constants

Titlu = "Logica computationala"

Formula = 3.0 - 3.0/5*5

Pi = 3.1415926

/* Identifierul unei constante nu este dependent de tipul literei. Pi și PI este același lucru . În clauze trebuie exprimate cu litere mici spre a le deosebi de variabile*/

IX.2. Predicate predefinite. Tipuri de obiecte: Simple. Compuse

Secțiunea **domains** definește tipuri utilizator cu ajutorul domeniilor de bază. Domeniile de bază (care nu mai trebuie specificate) sunt: **char** (caracter între apostrofuri), **integer** (întreg cu semn pe 16 biți), **real** (numere zecimale cu modulul între 1e-307 și 1e308), **string** (șir de caractere între ghilimele), **symbol** (șir de litere, cifre, underscore (_), primul caracter fiind o literă mică sau un sir ca la string, când trebuie introduse și spații). Datele de tip string și symbol sunt reprezentate intern în mod diferit.

Domeniul **char** conține și caractere escape precum: '\n', '\t'. Predicatul nl este un analog pentru write('\n').

Functii aritmetice

O expresie aritmetică PROLOG este o expresie uzuală cu operanzi uniți prin operatori. Operanzii pot fi numere sau variabile, iar operatorii sunt: +, -, *, /, div, mod și paranteze. Expresia apare în dreapta operatorului =, iar în stânga apare fie o variabilă nelegată căreia i se atribuie valoarea expresiei , fie o valoare cu care se compară valoarea expresiei.

Un număr hexazecimal e precedat de \$.

\$ FFF = 4095

Expresia este calculabilă dacă toate variabilele ce apar în ea (când apar) sunt legate. Operațiile sunt executate respectând prioritatea operanzilor. Tipul rezultatului depinde de tipul operanzilor și al operatorilor.

Tipul rezultatului:

Operand 1	Operator	Operand 2	Rezultat
Integer	+,-,*	integer	integer
Real	+,-,*	integer	real
Integer	+,-,*	real	real
Real	+,-,*	real	real
Integer or real	/	integer or real	real

Ordinea de efectuare a operațiilor:

Ordinea de efectuare a operațiilor este cea clasică:

Operator	Prioritate
+,-	1
* / mod div	2
- + (unar)	3

Operatori relationali:

Alți operatori sunt operatorii relaționali. Ei pot compara nu doar expresii numerice, ci și caractere (se compară codurile ASCII).

<	mai mic
<=	mai mic sau egal
=	egal
>	mai mare
>=	mai mare sau egal
<> sau ><	diferit

Functii și predicate PROLOG:

Functiile și predicatele PROLOG sunt:

Functie	returnează
X mod Y	restul împărțirii lui X la Y:
X div Y	partea întreagă a lui X împărțit la Y
abs(X)	valoare absolută a lui X
cos(X)	cosinus de X
sin(X)	sinus de X
tan(X)	tangentă de X
arctan(X)	arctangentă de X
exp(X)	e la puterea X
ln(X)	logaritm natural din X
log(X)	logaritm în baza 10 din X
sqrt(X)	radical din X
random(X)	obține în X un număr aleator din intervalul [0, 1]
random(X,Y)	X dat, Y generat aleator $0 \leq Y < X$
round(X)	X rotunjit
trunc(X)	trunchiază X

Funcții de ieșire

Predicatul **write**

După cum ați remarcat deja, **write** este un predicat cu care putem scrie pe ecran orice tip de obiecte. Formatul general:

write (*arg1, arg2, ..., argn*) *(i, i, ... i) /* i = input (intrare) */*

Un obiect compus **ob = carte (autor, titlu)** îl putem scrie și cu **write (ob)**, dar și mai clar prin **ob = carte (A,T)** : write ("autor: „A,” titlu: „ T).

Pentru a scrie cât mai clar, putem utiliza texte explicative (exemplu: "autor") și alinieri, utilizând tabulare ('t') sau linie nouă ('\n') în loc de '\n' se poate folosi și predicatul nl (newline).

Exemplu:

Se scrie un tabel de forma:

Nume	prenume	obiect

datele fiind extrase prin unificare dintr-o bază de date.

domains

nume, prenume, obiect = symbol

predicates

are(nume, prenume, obiect)

scrie_baza

scrie_tabel

clauses

scrie_baza:-

```
    write ("-----"), nl,
    write ("Nume", 't', "prenume", 't', "obiect"), nl,
    write ("-----"), nl,
    scrie_tabel.
```

scrie_tabel:-

```
    are (Nume, Prenume, Obiect),
    write (Nume, 't', Prenume, 't', Salariu), nl, scrie_tabel.
```

scrie_tabel.

Predicatul **writeln**

Este analog lui write, în plus are un format după care se scrie. Formatul general este:

writeln(format, arg1,...,argn) *(i, i, ... , i),*

unde *format* precizează formatul pentru fiecare argument.

Formatele au forma (sintaxa tip printf din C):

% [-]m.pf

unde:

- ‘_‘ - Determină alinierea la stânga (implicit la dreapta). Aici [,] arată caracterul optional.
- m - Este numărul maxim de cifre pentru partea întreagă.
- p - Este numărul maxim de cifre pentru partea zecimală
- f: - Poate fi:
 - f – real în zecimal fixat;
 - e – real în notație exponentială (ex. 12.5e-5);
 - g – real scurt (implicit);
 - d – caracter sau întreg fără semn;
 - u – caracter sau întreg fără semn;
 - x – caracter sau număr hexazecimal;
 - C – caracter sau întreg dat prin cod ASCII;
 - R – referință la baza de date;
 - X – hexazecimal lung (string, referință la o bază de dată);
 - s – string sau simbol.

Atât write cât și writef creează pe dispozitivul de ieșire curent (stdout, CON).

Predicate de intrare

- readln(string)** (o)
Cu readln se citește o linie de text, terminată cu newline.
- readint(integer)** (o)
- readchar (char)** (o)
- readreal(real)** (o)
Citește un integer, un char, respectiv un real.
- readterm(domain,term)** (i,i)
Citește un termen dintr-o bază de date sau file (vom reveni).
- file-str (filetext,text)** (i,o)(i,i)
Citește dintr-un fișier un text, sau creează un fișier în care scrie un text dat (vom reveni).
- inkey(char)** (o)
Citește un caracter de la tastatură.
- keypressed**
Așteaptă până când se introduce un caracter de la tastatură.
- unreadchar(char)** (i)
Inserează caracterul în buffer-ul stației (vom reveni).

Toate predicatele de intrare lucrează pe dispozitivul de intrare curent (stdin, CON).

Funcții de lucru cu caractere și string-uri

Așa cum am văzut anterior, un predicat poate fi polimorfic, după cum unele din argumentele sale sunt de intrare sau de ieșire. Acest polimorfism poate fi controlat de “flow pattern”, specificarea variantelor admise pentru argumente. Cu **i** specificăm un argument de intrare și cu **o**, de ieșire (i- input, o- output).

Exemplu:

= (i,i) (i,o) (o,i)

Pentru (o,i) sau (i,o) are loc atribuire, pentru (i,i) are loc comparație, (o,o) nu e admis.

În continuare, descriem funcțiile și predicatele însotite de “flow pattern”:

frontchar (string, char, string1)

(i,o,o) (i,i,o) (i,o,i) (i,i,i) (o,i,i)

Primul caracter din string este char, restul string-ului este string1. Primul pattern desparte string în char și string1, ultimul construiește pe string din char și string1, celelalte sunt pattern-uri de verificare.

fronttoken (string, token, string1)

(i,o,o) (i,i,o) (i,o,i) (i,i,i) (o,i,i)

Extrage din string primul token (primul cuvânt, tipul va fi symbol), restul string-ului este string1.

frontstr (n,string, frontstring, endstring).

(i,i,o,o)

Desparte string în frontstring (primele n caractere) și endstring.

concat (string1, string2, string3)

(i,i,o) (i,o,i)(o,i,i) (i,i,i)

Concatenează string1 cu string2, rezultă string3.

str_len (string,n)

(i,o) (i,i) (o,i)

n e lungimea lui string.

isname (string)

(i)

string e nume PROLOG (începe cu literă sau _, urmat de litere, cifre sau _).

format (outstring, format, arg1....argn)

(o,i,i,...)

“scrie” argumentele în outstring, respectând formatul dat.

Predicate de conversie

char_int(char,integer)

(i,o) (o,i) (i,i)

Convertește char în int, invers sau verifică o egalitate de coduri.

str_char (string,char)

(i,o) (o,i) (i,i)

Transformă un string de un singur caracter într-un caracter.

str_int (string, integer)

(i,o) (o,i) (i,i)

Transformă un string într-un caracter.

str_real (string,real)

(i,o) (o,i) (i,i)

Transformă un string într-un real.

upper_lower (string, string1)

(i,o) (o,i) (i,i)

Transformă literele mici în mari și invers.

upper_lower (char, char1)

(i,o) (o,i) (i,i)

Transformă literele mici în mari și invers.

Obiecte compuse definite cu ajutorul functorilor

Functorul seamănă cu predicatul, dar nu folosește la descrierea de fapte sau generalizări ale acestora prin reguli. Functorul permite definirea unui tip **record** (Pascal) sau **struct** (C). Mai precis, o parte a obiectelor compuse sunt de tip înregistrare.

Obiectele compuse se definesc în secțiunea domains, aşa cum erau definite și tipurile simple. Un obiect compus poate avea chiar și mai multe variante și se declară prin

domeniu = functor1(domeniu1, domeniu2, ...); functor2(domeniu1, domeniu2, ...);
unde domeniul este un domeniu predefinit sau definit de utilizator. Functor poate să nu aibă argumente.

Exemplificare:

domains

```
titlul, numele, prenumele = symbol
autorul = autorul(numele, prenumele)
obiect = carte (titlul, autorul);
fructe(symbol);
masina(symbol);
casa
virsta = i (integer); r(real); s(string)
posesor = symbol
```

predicates

```
are(posesor, obiect)
estate(posesor, virsta)
```

clauses

```
are("Andrei", carte("Prolog", autorul("Malita",
"Mihaela"))).
are("Andrei", carte("Algebra", autorul("Ion", "Ion"))).
are("Andrei", carte("Grafica in realitatea virtuala",
autorul("Ionescu", "Felicia"))).
are("Andrei", masina("Renault")).
are("Andrei", fructe("Mere")).
are("Andrei", casa).
estate("Andrei", i(35)).
```

Observați definirea obiectelor compuse cu mai multe nivele. Rețineți că este o mare diferență între fapte și obiecte. Ați văzut deja diferența dintre predicate și functori. Obiectul este definit prin functor, și nu prin predicat, aşa cum e definit faptul. De asemenea, faptele se pot generaliza la reguli, iar obiectele nu.

Operatorul “=” din zona **clauses** are dubla semnificație: de atribuire (când unul dintre operanzi este nelegat) și de comparare (când ambii operanzi sunt legați).

Alte domenii sunt: liste (tip*), ref, dbasedom, bt_selector, db_selector, place, file, reg, bgi_ilist etc.

IX.3. Mecanismul Backtracking. Predicatelor ! și fail

Procesul de căutare pornește de la începutul bazei de cunoștinte (fapte și clauze) pentru a se putea răspunde la întrebarea formulată. Într-o listă de predicate, verificarea prediciului curent pornește de la începutul listei. Dacă nu sunt soluții posibile se revine la prediciul anterior. Procesul de căutare a soluțiilor bazat pe revenire în vederea explorării tuturor alternativelor se numește *backtracking*. Dacă nu dorim toate soluțiile, ci numai unele, atunci “înghețăm” procesul de căutare pe anumite variabile și lăsăm mecanismul backtracking să lucreze pentru celelalte. Acest lucru este posibil folosind mecanismul tăieturii (eng. *Cut*), redat prin !. Acest mecanism se utilizează și atunci când forțăm obținerea doar a unei singure soluții. Dacă vrem ca procesul să continue în maniera nedeterministă (pentru explorarea tuturor alternativelor) se utilizează prediciul fail.

X. FISIERE ȘI BAZE DE DATE ÎN PROLOG

X.1. Fișiere

Fișierele, ca în alte limbi de programare, au un nume (recunoscut de sistemul de operare) și un nume simbolic (în cadrul programului) cu care e manipulat. Numele simbolic trebuie declarat în domains:

file = numesimbolic;....

Există nume simbolice standard ce nu mai trebuie să fie declarate.

Ele sunt:

com1 – ieșire la port serial;

- keyboard – citire de la tastatură (implicit);
- printer – ieșire la port paralel;
- screen – ieșire pe ecran;
- stdin – citire de la dispozitivul standard de intrare;
- stdout – ieșire la dispozitivul standard de ieșire;
- stderr – fișier standard de erori.

Asupra fișierelor se pot face mai multe tipuri de operații:

Deschiderea fișierelor

1. *openread* (simbolic, nume_DOS) (i,i)
Deschide un fișier pentru citire; dacă nu există va fi afișat un mesaj de eroare.
2. *openwrite* (simbolic, nume_DOS) (i,i)
Creează un fișier; dacă există, vechiul fișier este șters.
3. *openappend* (simbolic, nume_DOS) (i, i)
Deschide un fișier pentru adăugare; dacă nu există, va fi afișat mesaj de eroare.

4. *openmodify*(symbolic, nume_DOS) (i, i)

Deschide un fișier pentru scriere/citire; dacă nu există, afișează mesaj de eroare. Se poate folosi în conjuncție cu *filepos* și se poate accesa direct articolul ce urmează a fi modificat.

5. *filemode* (symbolic, mod) (i, i)

Se declară tipul fișierului: 0 – text, 1 – binar.

Un fișier text este organizat pe linii. Fiecare linie se termină cu carriage return (ASCII 13) și linie feed (ASCII 10). La citire, aceste coduri sunt egale cu nl. Un fișier binar este citit cu *readchar*.

Închidere de fișier

6. *closefile* (symbolic) (i)

Fișierul privit ca dispozitiv

7. *readdevice* (symbolic) (i) (o)

Este declarat/testat dispozitivul de citire. Acesta, implicit e *keyboard*.

8. *writedevice* (symbolic) (i) (o)

Este declarat/testat dispozitivul de scriere. Acesta, implicit e *screen*. Aceste predicate pot fi folosite pentru redirectări de dispozitive.

Lucrul cu fișiere

9. *filepos* (symbolic, filepos, mod) (i,i,i) (i,o,i)

Pozitionează/returnează poziția curentă *filepos*, a cărui semnificație depinde de mod:

- mod 0 – *filepos* relativ la început;
- mod 1 – *filepos* relativ la poziția curentă;
- mod 2 – *filepos* relativ la sfârșit.

10. *eof*(symbolic) (i)

Testează dacă indicatorul de articol e pe EOF (End of File); dacă nu, se obține fail.

11. *flush*(symbolic) (i)

Golește buffer-ul intern către fișier.

12. *existfile* (nume_DOS) (i)

Testează existența fișierului. Returnează *fail* dacă nu există.

13. *deletefile*(nume-DOS) (i)

Sterge un fișier; dacă nu există, se dă mesaj de eroare.

14. *renamefile*(nume_DOS_vechi, nume_DOS_nou) (i,i)

Redenumește un fișier; dacă nu există, afișează mesaj de eroare.

15. *disk*(DosPath) (i) (o)

Schimbă/afișează calea.

X.2. Baze de date interne

Turbo Prolog utilizează termenul de bază de date internă pentru a desemna faptele care pot fi adăugate sau șterse în timpul executării programului. Trebuie observat caracterul dinamic al cunoașterii modelate prin fapte și reguli.

Declararea unei baze de date interne se face în secțiunea *database*. Această secțiune conține descrierea structurii faptului sau faptelor care o compun, respectiv numele prediciților/predicatelor și tipul argumentelor acestora.

Un program poate lucra cu mai multe baze de date interne. Acestea trebuie declarate în secțiuni *database* distincte, cu nume diferite. Dacă se utilizează o singură bază și nu se precizează numele său, aceasta primește numele implicit *dbasedom*.

În exemplul următor se utilizează două baze de date interne, prima fără nume (implicit *dbasedom*), iar cea de a doua, cu numele “dbc”. Fiecare dintre acestea conțin fapte a căror structură este descrisă în secțiunile *database* corespunzătoare.

```
domains
    nume, prenume, compartiment = string
    oras, strada, luna = string
    nr, zi, anul = integer
    adresa = adresa(oras, strada, nr)
    data_nasterii = data_nasterii(zi, luna, anul)
database
    angajat(nume, prenume, compartiment)
    persoana(nume, prenume, adresa, data_nasterii)
database dbc
    catalog(nume, prenume, adresa, integer)
```

În cadrul unui program, un anumit tip de fapte nu poate apărea decât într-o singură bază de date. Aceste fapte pot fi utilizate ca oricare altele, existând încă plus posibilitatea de a fi adăugate sau șterse în timpul executării programului.

Conținutul bazei de date interne poate fi memorat într-un fișier și poate fi restaurat plecând de la acesta, prin predicițele *save* și, respectiv, *consult*.

<i>save (nume_fișier)</i>	/*	(i)	*/
<i>save (nume_fișier, nume_baza_date)</i>	/*	(i, i)	*/
<i>consult (nume_fișier)</i>	/*	(i)	*/
<i>consult (nume_fișier, nume_baza_date)</i>	/*	(i, i)	*/

Prima formă a acestor predicate se utilizează atunci când programul include o singură bază de date internă, căreia nu i s-a atribuit un nume explicit. De asemenea, este posibilă adăugarea sau ștergerea de fapte prin intermediul consolei sau automat, din interiorul programului¹³.

Adăugarea de noi fapte se poate face prin predicatele *asserta*, *assert* și *assertz*, cu forma generală:

```
asserta (nume_predicat(arg1,...argn) [,nume_baza_date])
assertz (nume_predicat(arg1,...argn) [nume_baza_date])
assert (nume_predicat(arg1,...argn) [,nume_baza_date])      /*
                           (i,i) */
```

Aceste predicate acționează astfel: *asserta* adaugă faptul specificat înaintea tuturor faptelor cu același predicat, în timp ce *assert* și *assertz* adaugă faptul după acestea (la sfârșit).

Ștergerea faptelor se poate realiza cu predicatele *retract* și *retractall*, a căror formă generală este:

```
retract (nume_predicat (arg1,...,argn) [,nume_baza_date])      /*(i,i)
                           (o,i)*
retractall (nume_predicat(arg1,...argn) [,nume_baza_date])      /*(i,i)
                           (,j)*/
```

Predicatul *retract* șterge din baza de date primul fapt care poate fi unificat cu argumentul său. Specificarea numelui bazei de date nu este obligatorie, deoarece aceasta poate fi identificată după structura clauzei din primul argument. Indicarea ei este însă o bună soluție de evitare a eventualelor erori de tip. Folosind o variabilă liberă drept prim argument, pot fi șterse toate faptele, aşa cum se poate vedea în exemplul următor:

```
șterge_tot: - retract (X, baza_fapte), write (X), fail.
```

Predicatul *retractall* șterge toate faptele care pot fi unificate cu primul său argument. Folosind drept prim argument o variabilă anonimă, *retractall* șterge toate faptele din baza de date respectivă.

¹³ Fișierul în care este înregistrată o bază de date internă poate fi creat sau modificat și direct, cu ajutorul unui editor de texte, cu condiția respectării următoarelor restricții:

- sunt admise numai caractere minuscule; majusculele pot apărea numai în termeni încadrați între ghilimele;
- nu sunt admise spații, cu excepția cazului când sunt încadrate de ghilimele;
- faptele cu același predicat trebuie grupate;
- nu se admit linii libere sau comentarii;
- nu se admit alte simboluri decât ghilimele.

X.3. Baze de date externe

Alături de soluția stocării în fișiere, Turbo Prolog oferă o serie completă de predicate pentru crearea și gestionarea unei baze de date. Aceasta permite, pe de o parte, să se lucreze cu volume mari de date sau cu date având structuri complexe și, pe de altă parte, să se creeze o bază proprie de date, oferind mecanismele fundamentale de stocare și regăsire rapidă a datelor.

O asemenea bază de date este denumită *externă* (spațiul de memorare este în afara programului; pe disc), pentru a face distincție de ansamblul de fapte (aserțiuni) declarate în secțiunea *database*, care pot fi și ele actualizate prin predicate de tipul *assert* și *retract*, dar sunt memorate în spațiul de memorie internă rezervat programului. Cu toată asemănarea de terminologie, baza internă este o bază de fapte utilizabile nemijlocit în procesul de inferență logică, în timp ce baza de date externă operează cu structuri diferite, presupune efectuarea explicită de operații de scriere, citire, căutare, ștergere, iar conținutul său poate fi utilizat numai după ce datele necesare au fost aduse în spațiul de execuție al programului și au fost, eventual, transpusă în forma necesară.

Există trei domenii predefinite *db_selector*, *ref* și *bt_selector*. Ultimul domeniu este util în definirea arborelui atașat unei baze de date. Indexul unei baze de date externe este organizat sub formă de b-arbore. Lucrul cu B+ arbori nu face obiectul acestor lecții. Cei interesați pot consulta Zaharia¹⁴ și colectiv.

O bază de date ocupă, din punct de vedere fizic, un spațiu unic și poate fi plasată fie într-un fișier pe disc, fie în memoria volatilă, alături de program. Ultima soluție prezintă avantajul unei viteze de lucru mult mai mari, dar și dezavantajele limitării la capacitatea de memorie internă, iar pentru a evita pierderea bazei de date la terminarea programului este necesară copierea sa pe un suport de memorie externă.

Înregistrările memorate pot avea diverse structuri, cu condiția ca acestea să posede declarații de domeniu corecte, să fie, cu alte cuvinte, termeni Prolog valizi. De altfel, sistemul utilizează pentru a se referi la înregistrări chiar cuvântul rezervat *Term*.

Indiferent de structura sa, fiecarei înregistrări i se atribuie, în momentul stocării în baza de date, un număr de identificare unic, care servește drept identificator în cadrul spațiului de memorare.

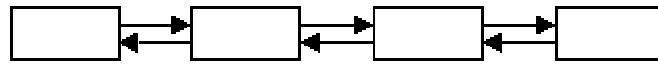
Fiecare înregistrare este plasată într-un lanț (eng. *chain*) printr-un sistem de legături. Fiecare lanț este identificat printr-un nume atribuit prin program. O bază de date trebuie să conțină cel puțin un lanț de înregistrări. De obicei, fiecare lanț conține înregistrări cu aceeași structură (domeniu identic), dar nu este obligatoriu să fie aşa. Din punct de vedere tehnic, este utilă reprezentarea dublu-înlățuită a listelor. De exemplu, dacă baza de date este compusă din tabelele *angajați* și *contracte*, lanțurile asociate pot fi reprezentate grafic astfel:

¹⁴ Zaharia D., Năstase P., Albescu F., Bojan I. – *Sisteme expert*, Societatea Știință și Tehnică S.A., 1998.

Lanțul “angajați”:



Lanțul “contracte”:



Crearea unei baze de date se face prin predicatul standard:

```
db_create (Id_bd, Nume_bd, Plasament) /* (i, i, i) */
```

unde:

- Id_bd este un identificator intern, obligatoriu de tipul predefinit *db_selector*;
- Nume_bd este numele bazei de date;
- Plasament, de tipul predefinit *Place*, indică locul în care se creează baza de date. Poate fi: *in_file*: baza de date este plasată pe disc, iar Nume_bd este, în acest caz, specificatorul fișierului respectiv; *in_memory*: baza de date este plasată în memoria internă.

Spre exemplu, declarația următoare creează o bază de date cu numele “date.pdb” aflată în catalogul “prolog” de pe discul “c”. Numele intern al acesteia este “pdb1”.

```
db_create (pdb1, "c:\prolog\date.pdb", in_file).
```

Ștergerea unei baze de date se realizează folosind predicatul *db_delete* cu forma:

```
db_delete (Nume_bd, Plasament) /*(i,i)*/
```

Exceptând cele două tipuri de operații prezentate, toate celelalte impun ca baza de date asupra căreia se acționează să fie deschisă¹⁵.

Deschiderea unei baze de date existente se face cu predicatul standard:

```
db_open (Id_bd, Nume_bd, Plasament) /* (i,i,i) */
```

iar închiderea acesteia, la terminarea prelucrării, prin predicatul:

```
db_close (Id_bd) /* (i) */
```

¹⁵ Se mai pot utiliza predicatele: *db_openinvalid(Id_bd, Nume_bd, Plasament)* pentru deschidere după avarie și *db_flush(Id_db)* pentru a termina scrierea, prin golirea zonei tampon (operație lentă.)

Copierea unei baze de date de pe un plasament pe altul se poate realiza cu predicatul:

db_copy (Id_bd, Nume_bd, Plasament_nou) /* (i,i,i) */

Reorganizarea spațiului de memorare pentru eliminarea zonelor rămase neutilizate ca urmare a ștergerilor de înregistrări sau îndeși se poate face folosind predicatul:

db_garbage collect (Id_bd) /* (i) */

Informații de ansamblu despre baza de date specificată se obțin folosind predicatul

db_statictis (Id_bd, Nr_înreg, Mem_int, Dim_bd, Spațiu_liber) /*(i,0,0,0,0)*/ instanțiiind argumentele cu următoarele valori:

- Nr_întreg indică numărul total de înregistrări existente (indiferent de tip);
- Mem_int indică memoria internă ocupată de tabelele care descriu baza de date;
- Dim_bd indică spațiul de memorie disponibil (pe disc sau în memoria internă, în funcție de plasamentul bazei de date).
- Spațiu_liber indică spațiul de memorie disponibil (în fișier sau în memoria internă, în funcție de plasamentul bazei de date).

Actualizarea presupune adăugarea de noi informații, modificarea celor existente sau ștergerea anumitor informații. Într-o bază de date Turbo Prolog, aceasta înseamnă adăugarea de noi înregistrări într-un lanț (chain), modificarea înregistrărilor existente sau ștergerea acestora.

Adăugarea de noi înregistrări se realizează prin predicatele:

chain_inserta(Id_bd, Lanț_înreg, Tip_înreg, Înregistrare, Nr_ref) chain_insertz(Id_bd, Lanț_înreg, Tip_înreg, Înregistrare, Nr_ref) /* (i,i,i,0)*/

Chain_inserta adaugă o înregistrare la începutul secvenței specificate, iar *chain_assertz* la sfârșitul său. Parametrii acestor predicate au următoarea semnificație:

- *Id_bd* - identificatorul bazei de date;
- *Lanț_înreg* - numele lanțului (chain) în care se adaugă înregistrarea;
- *Tip_înreg* - numele structurii (domeniului) înregistrării de adăugat;
- *Înregistrare* -înregistarea de adăugat;
- *Nr_ref* - numărul de referință atribuit înregistrării la adăugarea sa.

De asemenea, prin predicatul *chain_insertafter* se adaugă înregistrarea X imediat după cea cu numărul de referință dat prin *Pozitie*, iar referința asociată înregistrării adăugate este stocată în *Nr_ref*:

```
chain_insertafter(Id_bd, Lanț_înreg, Tip_înreg, Poziție, X, Nr_ref)
/* (i, i, i, i, 0) */
```

Alte predicate sunt: term_replace, term_delete, chain_terms, db_chains, chain_delete.

Predicatul *term_replace* înlocuiește înregistrarea cu numărul *Nr_ref* cu o altă *înregistrare* de tipul *Tip_înreg*:

```
term_replace(Id_bd, Tip_înreg, Nr_ref, Înregistrare) /* (i, i, i, i) */
```

Predicatul *term_delete* șterge înregistrarea cu numărul *Nr_ref* din *Lanț_înreg*:

```
term_delete(Id_bd, Lanț_înreg, Nr_ref) /* (i, i, i) */
```

Prin *chain_terms* se extrage termenul curent dintr-un lanț. Are caracter nedeterminist. Prin backtracking se obțin toți termenii lanțului. Forma predicatului este:

```
chain_terms(Id_bd, Lanț_înreg, Tip_înreg, Înregistrare, Nr_ref) /* (i, i, i, o, o) */
```

Predicatul *db_chains* permite obținerea, în ordine alfabetică, a numerelor lanțurilor dintr-o bază dată:

```
db_chains(Id_bd, Lanț_înreg) /* (i, o) */
```

Suprimarea unui lanț împreună cu toate înregistrările componente se realizează cu ajutorul predicatului *chain_delete*, de forma:

```
chain_delete(Id_bd, Lanț_înreg) /* (i, i) */
```

Pentru parcurgerea termenilor unui lanț se utilizează predicatele *chain_first*, *chain_last*, *chain_next* și *chain_prev*:

Predicatul *chain_first* returnează prima referință din lanțul de înregistrări specificat:

```
chain_first(Id_bd, Lanț_înreg, Nr_ref) /* (i, i, o) */
```

Predicatul *chain_last* returnează ultima referință din lanțul de înregistrări specificat:

```
chain_last(Id_bd, Lanț_înreg, Nr_ref) /* (i, i, o) */
```

Folosind legătura înainte, predicatul *chain_next* returnează următoarea referință după cea specificată:

```
chain_next(Id_bd, Referința_curentă, Următoarea_referință) /* (i, i, o) */
```

Folosind legătura înapoi, predicatul *chain_prev* returnează referința anterioară celei specificate:

```
chain_prev(Id_bd, Referința_curentă, Referința_anterioară) /* (i, i, o) */
```

Cunoașterea unei referințe face posibilă cunoașterea conținutului înregistrării folosind predicatul *ref_term*:

```
ref_term(Id_bd, Tip_înreg, Nr_ref, înregistrare) /* (i,i,i,o) */
```

BIBLIOGRAFIE

1. Albeanu G., *Algoritmi și limbaje de programare*, Editura Fundației România de Mâine, București, 2000.
2. State L., *Introducere în programarea logică*, Editura Fundației România de Mâine, București, 2004.
3. Clocksin W.F., Mellish C.S., *Programming in Prolog*, Springer-Verlag, Berlin, 1981.
4. Meszaros J., *Turbo Prolog 2.0, Ghid de utilizare*, Editura Albastră, Cluj-Napoca, 1996.
5. Malița M., *Antrenamente Prolog*, Editura Universității din București, 2000.
6. Zaharie D., Năstase P., Albescu F., Bojan I., *Sisteme expert*. Societatea Știință și Tehnică S.A., București, 1998.
7. Novikov P.S., *Elemente de logică matematică*, București, 1966.
8. Enescu Gh., *Logica simbolică*, Editura Științifică, București, 1971.
9. State L., *Elemente de logică matematică și demonstrarea automată a teoremelor*, Editura Universității București, 1989.
10. **DEX'1996**, *Dicționarul explicativ al limbii române*, Editura Univers Enciclopedic, București, 1996.

REȚELE DE CALCULATOARE

Lect. univ. MARIANA POPA

I. NOTIUNI DE BAZĂ

Construirea rețelelor de calculatoare a cunoscut o explozie nemaiîntâlnită. Rețelele de calculatoare au apărut din nevoia oamenilor de a partaja resurse fizice (imprimante, scannere etc.) și informații:

- Partajarea resurselor fizice aduce multe economii companiilor mari, în care spre exemplu, nu mai este necesară cumpărarea câte unei imprimante pentru fiecare sistem din rețea, ci o aceeași imprimantă este utilizată de către toți angajații.
- Partajarea informațiilor însă aduce pe lângă economiile materiale rezultate din costurile de transport și a consumabilelor și mari economii de timp, fiind permis accesul instantaneu la serverul cu aplicații, baze de date etc. Datele sunt memorate pe calculatoare numite servere și sunt accesate de angajați de la calculatoarele de birou, numite clienti.
- Nevoia de a comunica cât mai rapid a dus la apariția: poștei electronice (e-mail) folosită pentru comunicațiile zilnice, a videoconferințelor, a comerțului electronic (e-commerce), a camerelor de discuții (chat-room-ul), a mesageriei instantanee.
- Nevoia de divertisment a fost satisfăcută prin video la cerere, filme, muzică, prin jocuri pentru mai multe persoane cu simulare în timp real, prin realitate virtuală globală și partajată.

În multe situații calculatoarele de birou fixe au fost înlocuite cu calculatoare portabile, numite pe scurt PDA-uri (Personal Digital Assistant). Pe apă, în aer sau pe uscat, ori unde s-ar afla, posesorul unui PDA dorește să poată primi, trimite apeluri telefonice, să poată naviga pe Web sau să acceseze o bază de date. Așa s-au dezvoltat rețelele fără fir. Acestea au cunoscut importante aplicații în domeniul militar, în taximetrie, poliție, în telefonia fără fir și comerțul mobil (M-commerce).

Prin **rețea** de calculatoare înțelegem o colecție interconectată de calculatoare autonome.

Interconectarea (posibilitatea de a schimba informație) se face prin cabluri de cupru, fibre optice, microunde sau sateliți de comunicație.

Calculatoarele din rețea fiind autonome, între ele nu există relații de tip master/slave sau facilități de control al unei asupra altora din rețea.

Nu trebuie confundată o rețea de calculatoare cu un sistem distribuit în care existența mai multor calculatoare autonome este transparentă pentru utilizator, dar care nu e conștient de existența mai multor procesoare, sistemul arătând ca un singur procesor virtual. Într-un astfel de sistem, pentru execuția unui program, SO alege procesorul cel mai potrivit, identifică și transferă toate fișierele necesare către respectivul procesor și în final depune rezultatele în locul potrivit. Aici nimic nu este explicit, totul realizându-se automat fără cunoștință utilizatorului.

Într-o rețea de calculatoare fiecare utilizator se conectează explicit la un anumit calculator, comandă explicit execuția proceselor la distanță, transferă explicit fișierele și, într-un cuvânt, personalizează întreaga administrare a rețelei.

Diferență majoră între un sistem distribuit și o rețea de calculatoare nu apare la nivel de echipamente, ci la nivel de SO, unde se decide cine face munca: **sistemul sau utilizatorul**.

În proiectarea unei rețele de calculatoare (RC), deosebit de importante sunt:

- tehnologia de transmisie,
- scara la care operează rețeaua.

Există două tipuri de tehnologii de transmisie: rețele cu difuzare și rețele punct-la-punct.

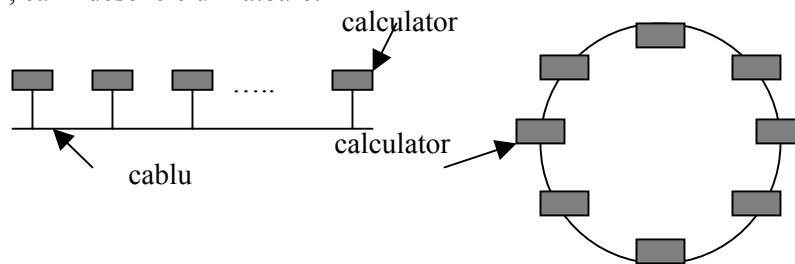
În **rețelele cu difuzare** există un singur canal de comunicație partajat de toate mașinile din rețea, pe care fiecare mașină poate trimite mesaje scurte, numite și **pachete**. Există un câmp de adresă pe pachet care precizează mașina căreia îi este adresat pachetul. Toate mașinile din rețea primesc pachetul, controlează câmpul de adresă, și păstrează pachetul numai acea mașină căreia îi este adresat, restul ignorându-l. Rețelele de dimensiuni mai mici folosesc această tehnologie.

În **rețelele punct-la-punct** există diferite conexiuni între mașini individuale, din care se formează trasee multiple pe care pachetele ajung de la sursă la destinație, trecând uneori prin mai multe mașini intermediare. În acest caz un rol deosebit îl joacă algoritmi de dirijare. Rețelele de dimensiuni mari folosesc această tehnologie.

În ce privește scara la care operează rețeaua, există următoarele tipuri de **sisteme cu procesoare multiple**:

- **Mașini de tip flux de date** - cu mai multe procesoare pe aceeași placă de circuite, situate la distanțe în jur de 10 cm. Acestea sunt calculatoare cu grad ridicat de paralelism, având mai multe unități funcționale ce pot lucra la același program.
- **Multicalculatoare** - cu procesoare în același sistem, situate la o distanță în jur de 1 m care comunică transmitând mesaje prin magistrale foarte scurte și rapide.
- **Rețelele locale** (Local Area Networks-LAN-uri) cu procesoare situate în același cameră, clădire sau campus, la distanțe de până la 10, 100 și respectiv 1000 m și care comunică prin schimb de mesaje prin cabluri mai lungi. Conectează calculatoare personale și stații de lucru cu scopul de a partaja resurse și de a schimba informații. Se disting de alte tipuri de rețele prin mărime, tehnologia de transmisie și topologie. LAN-urile au dimensiuni reduse, folosesc o tehnologie de transmisie ce constă dintr-un singur cablu la care sunt atașate toate mașinile și funcționează la viteze între 10Mbs și 100Mbs cu întârzieri de ordinul zecilor de microsecunde și cu erori foarte puține.

LAN-urile cu difuzare au două tipuri de tehnologii: cu magistrală sau cu inel, ca în desenele următoare:



- În **LAN-urile cu magistrală**, la fiecare moment unul din calculatoare poate transmite, iar celelalte aşteaptă. În cazul transmisiilor simultane este necesar un mecanism de arbitrage care poate fi centralizat sau descentralizat. În cazul centralizat există o unitate centrală de arbitrage a magistralei care decide cine urmează la rând pe baza unui algoritm intern care analizează cereri de transmisie. În cazul descentralizat nu există această unitate centrală și fiecare mașină hotărăște singură dacă să transmită sau nu controlul Ethernet(IEEE802.3)) este o rețea cu magistrală și control descentralizat în care calculatoarele transmit oricând doresc cu 10Mbs sau 100Mbs și dacă au loc ciocniri de pachete fiecare calculator aşteaptă un timp aleator și apoi încearcă din nou.
- În **LAN-urile cu inel** fiecare bit se propagă independent de ceilalți pe inel, fără să aștepte restul pachetului. Își aici este nevoie de o regulă pentru a arbitra accesele simultane la inel (exemplu IEEE 802.5 este un LAN de tip inel cu jeton ce operează la 4Mbs și la 16Mbs).

În funcție de modul de alocare a canalului, rețelele cu difuzare pot fi **statice** sau **dinamice**. În cele statice se divizează timpul în intervale discrete și se rulează un algoritm de alocare round-robin (RR), lăsând fiecare mașină să emită numai atunci când îi vine rândul. Dacă o mașină nu are nimic de transmis în cuanta de timp alocată, se irosește inutil capacitatea canalului și de aceea majoritatea sistemelor alocă la cerere canalul, adică în mod dinamic. În rețelele dinamice metodele de alocare sunt centralizate sau descentralizate.

Există și LAN-uri cu linii punct-la-punct în care linii individuale leagă o mașină specificată de o altă mașină specificată. Un altfel de LAN constituie un WAN în miniatură.

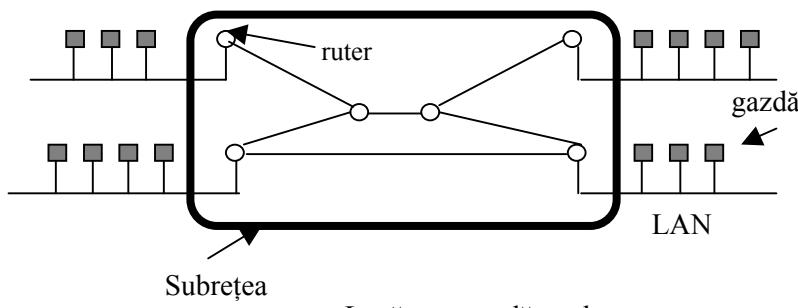
- **Rețelele metropolitane** (Metropolitan Area Networks-MAN-uri) cu procesoare situate în același oraș, la o distanță de cel mult 10km, sunt versiuni extinse ale LAN-urilor și deci folosesc tehnologii similare, putând suporta pe lângă date, voce și legături cu rețeaua locală de televiziune prin cablu.
- **Rețelele larg răspândite geografic** (Wide Area Networks – WAN-uri) cu procesoare în același țară sau pe același continent, situate la o distanță de 100 respectiv 1000km, dispun de o subrețea de comunicație compusă din **linii de transmisie** numite și **circuite, canale sau trunchiuri** și din **elemente de comutare** numite generic **rutere** ce pot fi noduri de comutare a pachetelor, sisteme intermediare sau comutatoare de date. La subrețea sunt legate calculatoare numite **gazde**. În general gazdele sunt conectate la un LAN, dar uneori pot fi legate direct de un ruter.

Linile de comunicație transferă bițiî între calculatoare.

Ruterele sunt calculatoare specializate ce conectează două sau mai multe linii de comunicație. La sosirea unor date pe o linie la router, acesta trebuie să aleagă o nouă linie pentru a transmite datele mai departe. Sarcina subrețelei este să transporte mesajele de la o gazdă la alta, acestea fiind conectate prin subrețeaua de comunicație.

În WAN-uri subrețeaua poate fi cu **comutare de pachete** (punct-la-punct sau memorează și transmite) sau cu difuzare. În primul caz subrețeaua conține numeroase cabluri sau linii telefonice, fiecare legând o pereche de rutere. Dacă

două rutere ce nu împart același cablu doresc să comunice, folosesc rutere intermedie în care pachetele trimise sunt primite în întregime, reținute un timp (timp în care linia de ieșire cerută devine liberă) și apoi transmise.



Legătura: gazdă - subretea

În cazul rețelelor cu comutare de pachete, topologia de interconectare a ruterelor poate fi de diverse tipuri: stea, inel, arbore, completă, inele intersectate, neregulată. În cel de al doilea caz, subretea folosește un satelit sau un sistem radio. În această situație fiecare ruter are o antenă cu care poate recepționa și transmite. Toate ruterele pot auzi semnalul de la satelit și în unele cazuri pot auzi și transmisia de la rutere către satelit.

Rețelele fără fir, sunt de trei tipuri:

- **Interconectarea componentelor unui sistem** (mouse, tastatură, cameră digitală etc): prin unde radio cu raza mică de acțiune, numită Bluetooth.
- **LAN-uri fără fir** se folosesc în clădirile în care cablarea este incomodă sau în spațiile în care se instalează greu o rețea. Ele au modem radio și antenă pentru a putea comunica între sisteme.
- **Wan-uri fără fir** au fost implementate prin rețeaua radio, și utilizate pentru voce și date.
- **Inter-rețele**. Dacă se pune problema ca persoane conectate la rețele diferite, uneori incompatibile, să comunice între ele, apare necesitatea conectării între ele. Acest lucru se face prin intermediul unor mașini speciale numite **porti**, care realizează conectarea și asigură translatăriile necesare atât la **nivel de hardware** cât și la **nivel de software**. O colecție de rețele interconectate se numește **interrețea** sau **INTERNET**, ea are procesoare și la o distanță de 10000km. O interrețea poate fi gândită ca o colecție de LAN-uri conectate printr-un WAN.
- **Rețelele casnice** sunt instalate o dată cu construcția clădirilor și fac accesibilă comunicarea între toate dispozitivele din casă (calculator, TV, DVD, combină muzicală, telefon fix sau mobil, frigider, cuptor, aparat de aer condiționat etc) prin Internet.

Rețelele sunt organizate pe **straturi** sau niveluri, fiecare nivel oferind servicii nivelurilor superioare. Nivelul n de pe o mașină **comunică**, folosind anumite reguli și convenții numite **protocole**, cu nivelul n de pe altă mașină.

Protocolul este deci o înțelegere între părți asupra modului de realizare a comunicării. În realitate nu se transmit direct informații între nivelul n al celor două mașini, ci datele de pe nivelul n se transmit nivelului n-1, de aici nivelului n-2 și aşa mai departe până la nivelul 1 sub care se află **nivelul fizic** prin care se produce comunicarea efectivă între cele două mașini.

Între două niveluri adiacente există o **interfață** care definește operațiile și serviciile primitive oferite de nivelul n și nivelul n+1.

O colecție de niveluri și protocoale de comunicație se numește **arhitectură de rețea**.

O listă de protocoale folosită de un sistem, câte un protocol pentru fiecare nivel se numește **stivă de protocoale**.

Să presupunem că la nivelul 5 o aplicație produce un mesaj M . Aceasta este furnizat nivelului 4 .

Nivelul 4 inserează un **antet** în fața mesajului, și trimite rezultatul nivelului 3. Antetul cuprinde o serie de informații numite **informații de control**.

Nivelul 4 nu impune limita de mărime a mesajelor , dar există o limită impusă de protocolul nivelului 3.

Nivelul 3 sparge mesajele în unități mai mici, numite **pachete**, și atașează fiecarui pachet un antet specific nivelului 3. Nivelul 3 trimite pachetele nivelului 2.

Nivelul 2 adaugă la fiecare pachet un antet și o încheiere, și trimite unitatea rezultată nivelului 1 care o va transmite fizic mașinii receptoare.

Mașina receptoare trimite mesajul în sus, din nivel în nivel, la fiecare nivel fiind eliminat antetul corespunzător.

Se observă din acest exemplu că, comunicarea între nivelul n de pe o mașină și nivelul n de pe mașina destinație nu are loc pe orizontală prin protocolul nivelului n cum ar părea la prima vedere.

La proiectarea nivelurilor unei rețele pot apărea o serie de probleme:

- Protocolul trebuie să determine la câte canale logice corespunde o conexiune și care sunt prioritățile acestora, deoarece există rețele cu două canale logice pe conexiune (unul este folosit pentru date normale iar celălalt pentru date urgente), astfel datele pot circula într-un singur sens - comunicare **simplex**; în ambele sensuri dar nu simultan - comunicare **semi-duplex** sau în ambele sensuri simultan - comunicare **duplex**.
- Fiecărui nivel îi trebuie un mecanism care să identifice emițătorii și receptorii .
- Este necesar un control al erorilor pe circuitele fizice care nu sunt perfecte. Același cod detector și corector de erori trebuie folosit de ambele capete ale unei conexiuni și în pus receptorul trebuie să poată anunța emițătorul ce mesaje nu au fost primite corect.
- Protocolul trebuie să furnizeze explicit receptorului informația necesară reconstituirii ordinii corecte a fragmentelor pentru că nu toate canalele păstrează ordinea mesajelor trimise.
- O altă problemă ce intervene la fiecare nivel se referă la **controlul fluxului de date**, adică la evitarea situației în care un emițător rapid trimite unui receptor lent date la viteză prea mare.
- Nu toate procesele ce se desfășoară în cadrul nivelurilor acceptă mesaje de lungime arbitrară, astfel de multe ori este necesarădezasamblarea mesajelor, transmiterea și apoireasamblarea lor.
- Atunci când este prea costisitoare alocarea unei conexiuni separate pentru fiecare pereche de procese comunicante, nivelul implicat în comunicare poate utiliza aceeași conexiune pentru mai multe conversații diferite. Această operație se numește **multiplexare** și **demultiplexare**, ea se realizează transparent și poate avea loc la orice nivel.

- Dacă transmiterea mesajului de la sursă la destinație se poate face pe mai multe căi, trebuie ales un anumit drum. Această problema se numește **dirijare sau rutare**.

Elementele active ale unui nivel se numesc **entități**. Entitățile acelaiași nivel dar de pe mașini diferite se numesc **entități pereche**. Dacă entitățile nivelului n implementează un serviciu folosit de nivelul n+1, nivelul n se numește **furnizor de servicii**, iar nivelul n+1 **utilizator de servicii**. Entitățile pot fi:

- entități software (un proces),
- entități hardware (un cip I/E inteligent).

Punctele nivelului n prin care nivelul n+1 are acces la serviciile oferite de acesta se numesc SAP-uri (**Service Acces Points - puncte de acces la servicii**).

Există două tipuri de servicii oferite de fiecare nivel nivelurilor superioare: - servicii orientate pe conexiuni și servicii fără conexiuni.

Serviciul orientat pe conexiuni este asemănător sistemului telefonic. Pentru a vorbi cu cineva:

- se ridică receptorul,
- se formează numărul,
- se vorbește,
- se închide.

Analog în serviciul orientat pe conexiuni:

- se stabilește o conexiune,
- se folosește conexiunea,
- se eliberează conexiunea.

Serviciul fără conexiuni este asemănător sistemului poștal, mesajele conțin adresele complete de destinație și fiecare mesaj circulă independent față de celelalte. Dacă două mesaje au aceeași destinație se poate întâmpla ca cel care a fost trimis al doilea să ajungă primul.

Una dintre caracteristicile unui serviciu este **calitatea serviciului**.

Un serviciu trebuie să fie **sigur**, adică să nu pierdă niciodată date. Pentru aceasta, receptorul trebuie să confirme primirea fiecărui mesaj, ceea ce duce la introducerea unui timp suplimentar și la întârzieri.

Servicii sigure orientate pe conexiuni pot fi:

- **Secvențele de mesaje**, în care este menținută delimitarea mesajelor. Dacă spre exemplu sunt trimise două mesaje de aceeași dimensiune, ele vor sosi la destinație sub forma a două mesaje distincte și niciodată ca un singur mesaj de dimensiune dublă.
- **Fluxurile de octeți**, în care nu există delimitări ale mesajelor. Dacă receptorul primește 2048 de octeți, el nu poate să dacă toți octetii provin de la același mesaj sau de la 2 sau mai multe mesaje mai mici.

Serviciile nesigure (neconfirmante) fără conexiuni: se mai numesc și **datagrame** și funcționează pe principiul **telegramelor**, unde nu se transmit confirmări către expeditor.

Se poate folosi și **serviciul datagrama confirmat** sau serviciul **cerere-răspuns**.

Formal, un serviciu este specificat printr-un **set de primitive** (operații), disponibile entității care folosesc acest serviciu. Aceste primitive comandă serviciului să execute anumite acțiuni sau să raporteze despre acțiunile executate de

o entitate pereche. Cele două tipuri de servicii, orientat pe conexiuni și fără conexiuni, au primitive diferite.

Exemple de primitive pentru implementarea unui serviciu simplu orientat pe conexiune:

- LISTEN (ascultă), blochează și așteaptă o conexiune;
- CONNECT (conectează) – stabilește conexiune cu o entitate pereche aflată în așteptare;
- RECEIVE (primește) – blochează în așteptare de mesaj;
- SEND (trimite) – trimite mesaj entității pereche;
- DISCONNECT (deconectează) – încheie conexiunea.

Serviciile și protocoalele sunt concepte distincte.

Prin **serviciu** înțelegem un set de primitive pe care un nivel le furnizează nivelului de deasupra. El spune ce operații poate oferi utilizatorilor săi fără să precizeze nimic despre modul de implementare al acestor operații. Serviciul este definit în contextul unei interfețe între două nivele.

Prin **protocol** înțelegem un set de reguli ce guvernează formatul și semnificația cadrelor, pachetelor sau mesajelor schimbate între ele de entitățile pereche ale unui nivel. Entitățile folosesc protocoale pentru a implementa operațiile serviciului lor.

Din cele două definiții se poate trage următoarea concluzie: serviciile sunt legate de interfețele dintre niveluri, iar protocoalele sunt legate de pachetele trimise între entitățile pereche de pe diferite mașini.

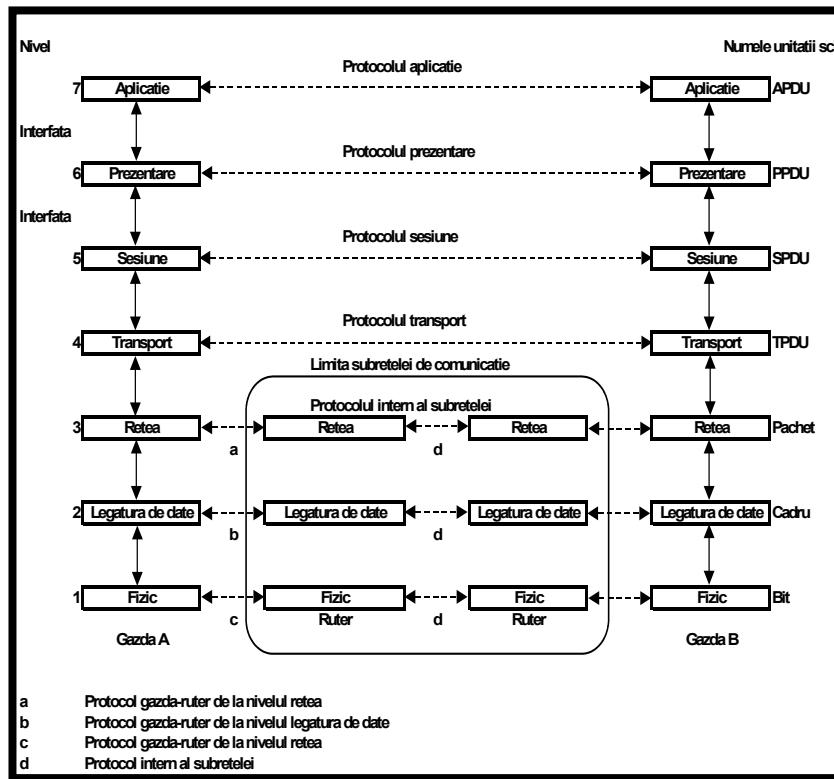
II. MODELE DE REFERINȚĂ

Arhitecturile de rețea cele mai cunoscute sunt **modelul de referință OSI** (Open System Interconnection – interconectarea sistemelor deschise) și **TCP** (Transmission Control Protocol).

Modelul OSI

Cuprinde 7 niveluri. La crearea lui stau următoarelor principii:

- Un nivel se creează când apare necesitatea unui nivel diferit de abstractizare.
- Fiecare nivel are un rol bine definit.
- Funcția fiecărui nivel trebuie aleasă încât să se poată defini protocoale standardizate.
- Delimitarea nivelurilor se face astfel încât să minimizeze fluxul de informații prin interfețe.
- Numărul nivelurilor trebuie să fie suficient de mare pentru a nu introduce funcții diferite în același nivel și trebuie să fie suficient de mic pentru ca arhitectura rețelei să fie funcțională.



Nivelul fizic are rolul de a transmite biții de la o mașină la alta printr-un canal de comunicație, variind câteva proprietăți fizice cum ar fi tensiunea și intensitatea curentului. Acest nivel trebuie proiectat în aşa fel încât să poată rezolva problemele tipice de tipul: cîți volți sunt necesari pentru reprezentarea unui bit 1 și respectiv a unui bit 0, dacă transmisia se poate realiza simultan în ambele sensuri ale canalului de comunicație, modalitatea de stabilire și de întrerupere a conexiunii, numărul de pini și utilitatea fiecăruia pin al conectorului de rețea. În funcție de acestea se modelează comportamentul semnalului. Pentru transmisie se pot utiliza mediul magnetic, cablul torsodat, cablul coaxial în bandă de bază, cablul coaxial de bandă largă, fibre optice sau comunicațiile fără fir (spectrul electromagnetic, transmisia radio, transmisia prin microunde, unde infraroșii și milimetrice, sistemul telefonic, rețeaua digitală cu servicii integrate de bandă largă).

Nivelul legătură de date

- are rolul de a transforma un mijloc de transmisie într-o linie disponibilă nivelului rețea fără erori de transmisie,
- obligă emițătorul să descompună datele în **cadre de date**, să transmită secvențial cadrele și să prelucreze **cadrele de confirmare** transmise de receptor,
- o alta funcție pe care o îndeplinește nivelul legătură de date este reglarea traficului, astfel încât un receptor lent să nu fie inundat de un emițător rapid.

- Emițătorul trebuie să dispună de mecanisme speciale care să îl informeze asupra spațiului tampon deținut de receptor la un moment dat,
- nivelul legătură de date cuprinde un subnivel de control al accesului la mediu care rezolvă problema controlului accesului la canalul partajat pentru rețelele cu difuzare.

Nivelul rețea are rolul de a controla funcționarea subrețelei. Acesta trebuie să detemine modul de dirijare a pachetelor de la sursă la destinație. Dirijarea se poate realiza prin intermediul tabelelor statistice (prin trasee care sunt stabilite la începutul fiecărei conversații) sau în mod dinamic (prin determinarea traseelor pentru fiecare pachet în parte în concordanță cu traficul din rețea la momentul respectiv).

Printre funcțiile nivelului rețea enumerăm: controlul congestiilor, tratarea întârzierilor, timpul de tranzitare, fluctuațiile. De asemenea, nivelul rețea se ocupă de transferul pachetelor dintr-o rețea în alta, în cazul în care apar complicații datorate modului de adresare sau a protocolelor diferite, dimensiunii prea mari a pachetelor etc. Pentru rețelele cu difuzare, dirijarea fiind simplă, nivelul rețea poate fi inexistent.

Nivelul transport descompune datele pe care le primește de la nivelul sesiune în unități mai mici, le trimite nivelului rețea și se asigură că acestea ajung corect. De asemenea, stabilește tipul de serviciu pe care îl furnizează nivelului sesiune și utilizatorilor rețelei.

În cadrul nivelului transport se realizează un schimb de informații între mașina sursă și mașina destinație prin intermediul unor programe similare instalate pe respectivele mașini, folosind antetele mesajelor și mesaje de control.

Nivelul sesiune are rolul de a facilita utilizatorilor de pe mașini diferite stabirea sesiuni între ei. Sesiunile trebuie să controleze dialogul, astfel încât utilizatorii să respecte regulile impuse comunicării, să gestioneze jetonul (în cazul în care doi utilizatori încearcă simultan o operație critică) și să asigure sincronizarea introducând puncte de control.

Nivelul prezentare are rolul de a verifica sintaxa și semantica informațiilor transmise, asigurând comunicarea între mașini care folosesc codificări diferite ale datelor.

Nivelul aplicație conține o multitudine de protocole utilizate frecvent cum ar fi HTTP (protocol de transmitere a hipertextului), FTP (protocol pentru transferul fișierelor), SMTP (protocol pentru poșta electronică) etc.

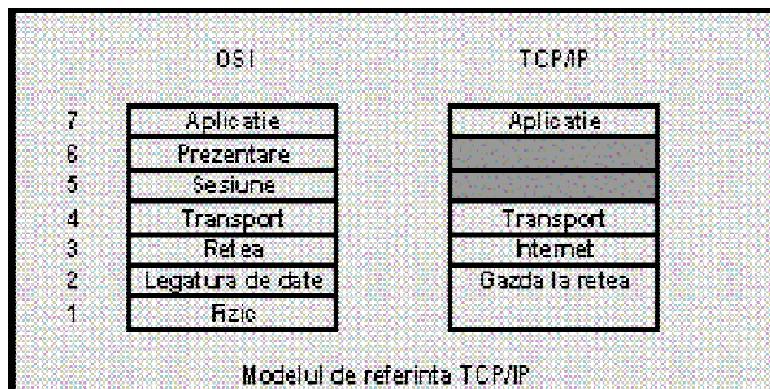
La transmiterea datelor în rețea din nivel în nivel sunt utilizate diferite dispozitive: repetoare, noduri, comutatoare, punți, rutere și porți.

Legătura între aceste dispozitive și datele transmis poate fi prezentată pe scurt astfel:

- **Repetoarele** sunt dispozitive analogice conectate între două segmente de cablu. Ele înțeleg doar tensiuni electrice, rolul lor este de a amplifica și trimite pe celălalt cablu un semnal apărut pe unul din cabluri.
- Un **nod** are mai multe linii de intrare pe care le unește din punct de vedere electric, el nu amplifică semnalul, cadrele ajuște pe o linie sunt trimise afară pe toate celelalte linii, nu recunoaște și nu utilizează adresele 802. Un nod formează un singur domeniu de coliziune. Nodurile suportă multe plăci de extensie cu mai multe intrări.

- **Comutatoarele** și **punțile** rutează cadre pe baza adreselor. Comutatorul conectează calculatoare individuale, fiecare port al comutatorului este conectat la un singur calculator. Punțile conectează numai rețele.
- **Ruterul** funcționează la nivelul rețea, interconectează mai multe rețele locale de tipuri diferite, care utilizează același protocol de nivel fizic. Când un pachet ajunge la un ruter, acesta elimină antetul și sfârșitul cadrului, apoi pachetul localizat în informația utilă a cadrului trece la software-ul de rutare care folosește antetul pachetului pentru a alege o linie de ieșire.
- **Portile** (de la nivelul transport) conectează două calculatoare ce utilizează diferite protocole de transport orientate pe conexiune, poate copia pachete de la o conexiune la alta, refăcând pachetele, dacă este nevoie.
- **Portile** (de la nivelul aplicație) înțeleg formatul și conținutul datelor și traduc mesajul de la un format la altul.

Modelul de referință TCP/IP



Modelul de referință TCP/IP a fost definit prima dată în 1974 de catre Cerf și Kahn, răspunzând cerințelor de interconectare a rețelelor prin satelit și radio. Prin acest model s-a dorit asigurarea conexiunilor și în cazul nefuncționării unor echipamente din rețea. De asemenea, s-a dorit asigurarea unor servicii foarte diferite inclusiv transferul de fișiere și transmiterea vorbirii în timp real.

Nivelul Internet este un nivel inter-rețea fără conexiuni, fiind cel mai important nivel al acestui model. El are rolul de a permite calculatoarelor să transmită pachete de date în orice rețea și să asigure transportul lor independent până la calculatorul destinație. În cadrul acestui nivel sunt definite un format de pachet și un protocol numit IP (Internet Protocol).

Nivelul internet este similar nivelului rețea din modelul OSI, având ca probleme principale dirijarea pachetelor și evitarea congestiei.

Nivelul transport are rolul de a permite conversații între utilizatorii a două calculatoare, respectiv sursă și destinație. Nivelul cuprinde două protocole capăt-la-capăt: protocolul TCP și protocolul UDP.

Protocolul **TCP** (Transmission Control Protocol) este un protocol sigur, orientat pe conexiune care realizează controlul transmisiei (pentru a elimina erorile de transmisie ce pot apărea între două mașini aflate în rețea) și controlul fluxului

(pentru a evita inundarea unui receptor lent de către un emițător cu o viteză mult mai mare).

Protocolul **UDP** (User Datagram Protocol) este un protocol nesigur și neorientat pe conexiune care oferă posibilitatea utilizatorilor să folosească propriul lor control al transmisiei și al fluxului. Protocolul asigură comunicarea rapidă client-server și între aplicații, fără să garanteze însă acuratețea.

Nivelul aplicație cuprinde toate protocolele de nivel înalt cum ar fi protocolul de terminal virtual (TELNET) care permite unui utilizator de pe o mașină să se conecteze și să lucreze pe o mașină situată la distanță, transferul de fișiere (FTP) care posedă un mecanism de mutare eficientă a datelor de pe o mașină pe alta, poșta electronică (SMTP), precum și alte servicii: serviciul numelor de domenii (DNS) care stabilește corespondența dintre numele gazdelor și adresele rețelelor, protocolul pentru transferarea de știri (USENET), protocolul folosit pentru aducerea paginilor de pe web (HTTP).

Nivelul gazdă-rețea are rolul de a conecta gazda la rețea, prin intermediul unui protocol. Protocolul nu este definit, fiind diferit de la gazdă la gazdă și de la rețea la rețea.

Putem face o **comparatie** între cele două modele:

Asemănări

- Protocolele sunt independente și stratificate.
- Nivelele îndeplineșc în linii generale aceleași funcții.

Deosebiri

- Modelul OSI are șapte niveluri, modelul TCP are patru.
- Spre deosebire de modelul TCP, OSI are clar delimitate concepte esențiale (servicii, interfețe, protocole), caracteristică modernă care îl aseamănă cu programarea orientată pe obiecte.
- Protocolele OSI sunt mai bine protejate și pot fi mai ușor înlocuite și adaptate la tehnologiile moderne decât în cazul modelului TCP.
- Modelul OSI suportă atât comunicația fără conexiuni, cât și pe cea orientată pe conexiuni în nivelul rețea, dar numai ultimul tip de comunicație în cadrul nivelului transport. Modelul TCP suportă numai primul tip de comunicație la nivelul rețea și ambele la nivelul transport.
- Modelul OSI (mai puțin nivelurile sesiune și prezentare) este extrem de utilizat pentru a discuta rețele de calculatoare, dar protocolele OSI nu s-au impus în lumea calculatoarelor.
- Protocolele modelului TCP/IP au devenit extrem de populare fiind larg utilizate.

Critici aduse modelului TCP/IP: nu are clar delimitate concepte de serviciu, interfață, protocol; nu este general și nu poate lucra cu altă stivă de protocole; nivelul gazdă-rețea nu este suficient dezvoltat; nu sunt delimitate în cadrul modelului nivelurile fizic și legătură de date, deși sunt complet diferite; protocolele sunt greu de înlocuit, fiind răspândite pe scară largă, de cele mai multe ori gratuit.

Critici aduse modelului OSI: ratarea momentului de apariție a protocolelor; tehnologii proaste de delimitare a nivelelor și de proiectare a protocolelor care sunt greu de implementat și ineficiente în funcționare; implementări proaste ale modelului și a protocolelor; politici proaste privind utilitatea și sfera de aplicabilitate a modelului.

III. EXEMPLE DE REȚELE

Rețelele se deosebesc prin istoric, administrare, facilități oferite, proiectare și prin **comunitățile** lor de utilizatori. În prezent funcționează în lume un număr foarte mare de rețele printre care:

- **INTERNET-ul**, colecție de rețele ce oferă anumite servicii comune și utilizează protocoale comune. Nu este controlat de nimeni și are o dezvoltare continuă. A apărut după 1 ianuarie 1983 când TCP/IP a devenit protocol oficial. De la apariția în 1983 a Internetului numărul calculatoarelor din rețea a crescut exponențial (în 1990 rețeaua Internet cuprindea 3000 rețele cu 20000 calculatoare, în 1992 a fost conectată gazda cu numărul 1000000, iar în 1995 în rețea existau mai multe coloane vertebrale, sute de rețele regionale, zeci de mii de LAN-uri, milioane de gazde și zeci de milioane de utilizatori. În fiecare an mărimea Internetului se dublează. O mașină este pe Internet dacă folosește stiva de protocoale TCP/IP, are o adresă IP și poate trimite pachete IP către toate celelalte mașini de pe Internet. Dezvoltarea Internetului din anii 1990 a dus la apariția ISP-urilor (Internet Service Provideri – furnizori de Internet).

Aplicații principale ale Internetului sunt:

Poșta electronică - mijloc de a interacționa cu lumea exterioară, depășeste telefonul și poșta obișnuită, programele de poștă electronică fiind disponibile pe orice tip de calculator.

Sistemele de poștă electronică pun la dispoziție 5 funcții de bază:

- **compunerea** (procesul de creare a mesajelor și a răspunsurilor),
- **transferul** (deplasarea mesajului de la autor la receptor. Aceasta necesită stabilirea unei conexiuni la destinație sau la o mașină intermediară, emiterea mesajului și eliberarea conexiunii),
- **raportarea** (informarea autorului despre ce s-a întâmplat cu mesajul (livrat, respins, pierdut)),
- **afișarea mesajelor primite** (utilizatorii își pot citi poșta),
- **dispoziția** (se referă la ce face receptorul cu mesajul după ce l-a primit (eliminare înainte de citire, aruncare după citire, salvare etc.)).

Grupurile de știri (USENET) - forumuri specializate în care utilizatorii cu un interes comun pot să facă schimb de mesaje. Fiecare grup de știri are eticheta, stilul și obiceiurile sale proprii.

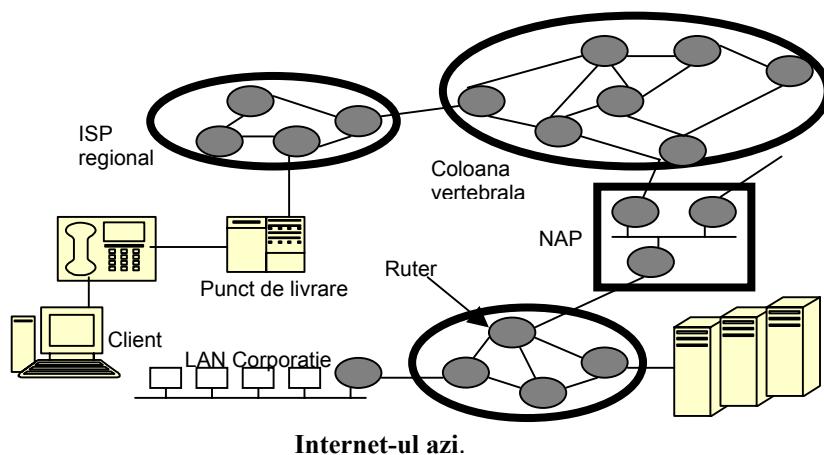
USENET este un sistem de grupuri de știri cu întindere planetară numit net news (rețea de știri). În zilele noastre o mare parte din trafic este transmis prin Internet. Unele site-uri Internet nu primesc știri iar altele primesc știri fără a fi Internet. Astfel USENET și Internet sunt lucruri distințe.

Un grup de știri este format dintr-un forum universal de discuții pe teme specifice. Cei interesați de o anumită temă pot subscrive la un grup de știri specific temei. Aceștia pot folosi un agent utilizator special pentru a citi toate mesajele trimise la un grup de știri, agent care este un program de citire știri. Fiecare articol trimis unui grup de știri este transmis automat tuturor celor ce formează grupul oriunde ar fi ei. Numărul grupurilor de știri este peste 10.000 și pentru ca ele să poată fi gestionate au fost organizate într-o ierarhie. Aici întâlnim:

Conectare la distanță (folosind Telnet, Rlogin sau alte programe). **Transfer de fișiere** de pe o masina din Internet pe alta folosind programul FTP.

WWW (World Wide Web) inventată de fizicianul **Tim Berners Lee**. Prin WWW site-urile pun la dispoziție utilizatorului pagini cu informații ce cuprind pe lângă text, poze, sunet, video, legături la alte pagini sau hărți etc.

Web-ul poate fi considerat o colecție imensă de documente numite pagini răspândite în toată lumea. Fiecare pagină poate conține link-uri (legături) la alte pagini aflate oriunde în lume. Se spune că aceste pagini folosesc hypertext. O legătură funcționează dând un clic pe ea și astfel se ajunge la pagina indicată de legătură. Procesul se poate repeta cât timp paginile accesate conțin link-uri. Paginile pot fi văzute prin programe de navigare numite **browsere** printre care amintim Mosaic și Netscape, care aduc pagina dorită, interpretează textul și comenzi de formatare conținute în text și afișează pagina formatată corespunzător pe ecran.



Internet-ul azi.

ARPANET-ul a apărut din nevoie de tehnologie utilă în scopuri militare, tehnologie care avea la bază o rețea cu comutare de pachete formată dintr-o subrețea și din calculatoare gazdă, și care în anii 1974 a culminat cu introducerea de către Cerf și Kahn a protocolului TCP/IP. În 1983 din ARPANET a derivat o subrețea separată numită MILNET. Conectarea la ARPANET a unui număr din ce în ce mai mare de LAN-uri a dus la crearea DNS-ului (Domain Naming System – Sistemul numelor de domenii) care organiza mașinile în **domenii** și punea în corespondență numele gazdelor cu adrese IP.

- **NSFNET-ul** își are originea în NFS (National Science Foundation – Fundația Națională de Științe din SUA), care la sfârșitul anilor '70 a văzut impactul ARPANET-ului asupra cercetării universitare și pentru a se putea conecta la ARPANET a organizat o rețea virtuală CSNET centrată în jurul unei singure mașini ce era suport pentru linii telefonice și care avea conexiuni cu ARPANET-ul și alte rețele. Prin CSNET cercetătorii puteau suna și lăsa poștă electronică pentru a fi citită ulterior de alte persoane.

În 1984 NFS a construit o coloană vertebrală care lega supercalculatoare din 6 orașe și **20 de rețele regionale**. Rețeaua obținută a fost numită NSFNET și ulterior a fost conectată la ARPANET. Își în Europa există rețele comparabile cu NSFNET cum ar fi **EBONE** care este coloana vertebrală IP pentru organizații de cercetare, sau **EuropaNET**, rețea orientată spre domeniul comercial.

Au apărut astfel NAP-urile (Network Access Point – punct de acces la rețea) care ajutau ca orice rețea regională să poată comunica cu orice altă rețea regională.

Rețele orientate pe conexiune

Există după cum știm două tipuri de subrețele: subrețelele fără conectare și subrețelele orientate pe conexiune. Susținătorii subrețelelor fără conexiune provin din comunitatea ARPANET/Internet, în care era admisă toleranță la defecte și nu se făcea taxarea clientilor.

Susținătorii rețelelor orientate pe conexiune provin din lumea comunicațiilor pe linii telefonice, pentru care calitatea serviciilor este importantă, iar facturarea este modul lor de supraviețuire. Într-o rețea fără conexiune, dacă la același ruter ajung mai multe pachete în același moment, ruterul va fi sufocat și va pierde din pachete. Dacă expeditorul observă le va retrimit, dar calitatea serviciilor va fi proastă, mai ales pentru comunicațiile audio sau video, excepție făcând cazurile în care rețeaua este doar foarte liberă.

X.25 - primele rețele orientate pe conexiuni **care** suportă circuite virtuale atât comutate, cât și permanente, au fost dezvoltate în anii '70 și au funcționat aproape un deceniu.

În anii '80, rețelele X.25 sunt înlocuite cu un nou tip de rețea, denumit **Frame Relay** (Releu de Cadre), rețea orientată pe conexiune, fără control al erorilor și fără control al fluxului de date. Frame Relay seamănă cu o rețea locală de dimensiuni mari și a fost utilizată la interconectarea diverselor rețele locale aflate în birourile companiilor. A avut un succes modest.

ATM (ATM Asynchronous Transfer Mode, rom: Mod de Transfer Asincron) - rețea orientată pe conexiune, în care spre deosebire de rețelele telefonice în care majoritatea transmisiilor sunt sincrone (strâns legate de un semnal de ceas), transmisiile sunt asincrone.

ATM-ul a rezolvat majoritatea problemelor legate de rețele și telecomunicații, prin unificarea transmisiilor de voce, date, televiziune prin cablu, telex, telegraf, într-un singur sistem integrat.

Cum rețelele ATM sunt orientate pe conexiuni, un apel presupune transmiterea unui mesaj pentru stabilirea conexiunii, trimiterea celulelor spre destinație pe același traseu, și sosirea acestora în ordinea transmiterii. Rețelele ATM sunt organizate ca WAN-uri cu linii și rutere ce lucrează la viteze de 155 Mbs și 622 Mbs. Ideea de bază în ATM este că se transmit toate informațiile în pachete mici, de dimensiune fixă, denumite celule (cells). Celulele au 53 de octeți, din care 5 octeți reprezintă antetul, iar restul de 48 reprezintă informația propriu-zisă.

Possibilitatea rețelelor ATM de a multiplica o celulă pe care o primesc la intrare, pe mai multe linii de ieșire, a fost speculată de distribuitorii TV care trebuiau să transmită un același program de televiziune către mai mulți receptori.

Cum ATM lucrează cu celule mici, nici o linie nu va fi blocată mai mult timp ceea ce garantează calitatea serviciilor. Livrarea celulelor nu este garantată, dar ordinea lor da, ceea ce face ca soluția ATM să fie mai bună decât cea oferită de Internet, unde pe lângă pierderea pachetelor și ordinea de ajungere a acestora la destinație poate fi oricare (nu are legătură cu ordinea de transmisie).

Modelul de referință ATM este un model tridimensional, compus din trei niveluri, două având și câte două subniveluri:

- **Nivelul fizic**, în care se planifică biții și se analizează voltajul, cu subnivelurile:
 - o **PMD** (Physical Medium Dependent, rom: dependent de mediul fizic) are drept scop transferul bițiilor și planificarea transmisiei la nivelul I.
 - o **TC** (Transmission Convergence, rom: convergența transmisiei). Rolul nivelului TC este să convertească fluxul de biți primiți de la PMD în flux de celule și să trimită celulele nivelului ATM.
- **Nivelul ATM** are drept scop celulele, stabilind structura, câmpurile și transportul acestora, tratează congestiile și modul de stabilire și eliberare al circuitelor virtuale.
- **Nivelul de adaptare ATM** numit și **AAL** (ATM Adaption Layer), permite utilizatorilor să trimită pachete mai mari decât o celulă, segmentează aceste pachete, transmite celulele individual și le reasamblează la celălalt capăt. El are două subniveluri:
 - o **SAR** (Segmentation And Reassembly, rom: segmentare și reasamblare) subnivel la care sunt descompuse pachetele în celule - la capătul la care are loc transmisia - și sunt recompozute la destinație.
 - o **CS** (Convergence Sublayer, rom: subnivel de convergență) oferă diverse tipuri de servicii : transfer de fișiere, video la cerere,etc.
- Plus orice alt nivel propus de utilizator.

Rețelele ATM nu s-au dezvoltat însă pe măsura așteptărilor.

ETHERNET-UL este cea mai populară dintre rețelele locale. Prin rețea Ethernet se transmit informații între calculatoare la viteze foarte mari. Standardul Ethernet este definit de IEEE (Institute for Electrical and Electronic Engineers) ca **IEEE 802.3**. Standardul definește regulile pentru configurarea unei rețele Ethernet precum și modul de interacțiune între diferitele elemente ale unei astfel de rețele. Ideea acestui standard este următoarea: stația care dorește să transmită, ascultă cablul:

- dacă este ocupat, așteaptă până se eliberează;
- dacă este liber, transmite imediat;
- dacă două sau mai multe stații încep să transmită simultan pe un cablu liber, apare coliziunea ;
- toate stațiile intrate în coliziune îintrerup transmisia, așteaptă o perioadă de timp aleatorie și repetă întregul proces de la capăt.

Fiecare calculator echipat cu o placă de rețea Ethernet, denumit și stație, funcționează independent de toate celelalte stații din rețea: nu există control centralizat. Toate stațiile atașate la rețea sunt conectate la același sistem de

transport pentru semnal, denumit mediu de comunicație. Informația este transmisă serial, un bit la un moment dat, prin linia de comunicație către toate stațiile atașate acesteia.

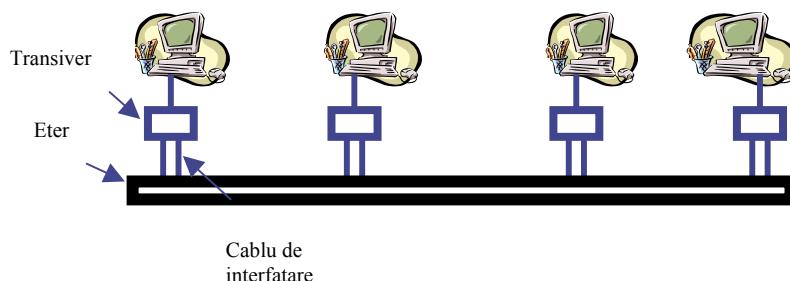
În **prima variantă**, Ethernets folosea sistemul ALOHANET cu transmisie radio pe două frecvențe:

- trimite (upstream – către calculatorul central);
- primește (downstream – de la calculatorul central).

Expeditorul transmitea un pachet care conținea datele pe canalul Trimite și prima răspuns pe canalul Primește. Dacă apăreau coliziuni pe canalul Trimite se retransmitea pachetul. Pe celălalt canal coliziunile erau excluse fiind un singur transmițător. Acest sistem era viabil numai în condiții de trafic redus pe canalul Trimite.

A **două variantă** a fost creată de Bob Metcalfe și colegul său David Boggs – care au realizat o rețea locală de calculatoare de până la 256 de calculatoare folosind ca mediu de transmisie cablul coaxial gros. Un cablu cu mai multe mașini atașate în paralel este numit **cablul multidrop**.

Ethernetul era mult îmbunătățit. El reușea să evite coliziunile prin **ascultarea canalului** (dacă se facea o transmisie pe canal, calculatorul care dorea să trimită date la un moment dat aștepta ca aceasta să se încheie și apoi transmitea și el), lucru posibil pe un cablu unic.



Aşa a apărut **Ethernet-ul Xerox**

Mai rămânea însă de rezolvat problema transmisiei simultane a două sau mai multe calculatoare.

Ethernet-ul Xerox a fost standardizat pentru o rețea Ethernet de 10Mbps, denumită și standardul **DIX**, și cu mici modificări acest standard a devenit în 1983 **IEEE 802.3**.

Alte standarde LAN cunoscute sunt : **Token Bus** (Jeton pe magistrala – 802.4) și **Token Ring** (Jeton pe Inel – 802.5).

Apariția calculatoarelor portabile a adus cu ea și ideea conectării la Internet fără fir, astfel a fost creat standardul **802.11** care lucrează atât în prezența unei stații de bază cât și în absența acesteia. Acest standard poate transmite un pachet IP într-un LAN fără fir la fel cum era transmis un pachet IP prin Ethernet. Conexiunea dintre sistemele 802.11 și lumea exterioară se numește **portal**. O variantă a standardului 802.11 este 802.11a care are o bandă de frecvență largă și poate depăși viteza de 54Mbps. Standardele 802.11 au multe variante și ele pot face INTERNET-ul mobil.

În august 2002 a fost finalizat **Standardul 802.16** care a aparut ca o soluție pentru serviciile de voce și de Internet de mare viteză. Ca și 802.11, 802.16 a fost proiectat pentru a oferi comunicații fără fir de mare viteză. Deși par asemănătoare, cele două standarde diferă semnificativ, astfel 802.16 oferă servicii pentru clădiri care nu sunt mobile și operează într-o gamă de frecvențe mult mai înalte, de 10-66 GHz etc.

Standardul IEEE 802.15 – Bluetooth aducea cu el un alt element de noutate, eliminarea cablurilor dintre anumite dispozitive.

Standardul IEEE 802.1Q sau VLAN (Virtual LAN) a apărut din necesitatea de a recabla cladirile în întregime doar cu ajutorul software-ului . VLAN-urile se bazează pe comutatoare dedicate. Ele pot avea însă și hub-uri la periferie.

IV. SECURITATE

Într-un mediu de rețea, trebuie să existe garanția că datele secrete sunt protejate, astfel încât numai utilizatorii autorizați să aibă acces la ele. Nu numai protejarea informațiilor confidențiale este importantă, ci și protejarea operațiilor efectuate în rețea.

Există patru amenințări majore la securitatea unei rețele:

- accesul neautorizat;
- alterarea electronică a datelor;
- furtul de date;
- daunele intenționate sau accidentale.

Aspectele teoretice ale securitatii sunt studiate de **criptografie**. Aceasta studiază tehniciile matematice care au legătură cu diferite aspecte ale securității informației, ca de exemplu confidențialitatea, integritatea datelor, autentificarea unei persoane sau a originii datelor.

Metodele de criptare sunt împărțite în două categorii:

- cifruri cu substituție (fiecare literă sau grup de litere se substituie cu altă literă sau alt grup de litere),
- cifruri cu transpoziție (reorganizează literele fără a le substitui cu altele).

Exemple de algoritmi de criptare utilizați de criptografia modernă:

- algoritmii cu cheie secretă (DES (Data Encryption Standard), AES (Advanced Encryption Standard) etc.),
- algoritmii cu cheie publică (RSA creat de Rivest, Shamir și Adelman)

Semnăturile digitale și gestionarea cheilor publice sunt de asemenea două teme de bază studiate în securitatea rețelelor

Din punct de vedere **practic**, principalul motiv care conduce la apariția unor breșe de securitate este configurarea greșită sau necorespunzătoare a sistemului victimă. Majoritatea sistemelor de operare sunt livrate într-o configurație

nesigură. Există două manifestări ale insecurității software-ului livrat ce pot fi clasificate ca:

- **starea activă a insecurității** (anumite utilitare de rețea, atunci când sunt activate, pot crea serioase riscuri de securitate, produsele software sunt de multe ori cu aceste utilitare activate necesitând interventia administratorului de rețea pentru dezactivarea sau configurarea lor corespunzător),
- **starea pasivă a insecurității** (sistemele de operare care au înglobate utilitare de securitate sunt eficiente doar când sunt activate, activare ce trebuie făcută în general tot de administratorul de rețea, în starea pasivă, aceste utilitare nu sunt niciodată active, deoarece de obicei utilizatorul nu știe de existența lor).

Instrumentele software utilizate în securitate sunt:

- **Scanerul**, program care detectează automat punctele slabe în securitatea unui sistem local sau la distanță. În securitatea Internetului, este cel mai important utilitar. Adevăratele scanere sunt scanere de porturi TCP, adică programe care atacă porturile și serviciile TPC/IP și înregistrează răspunsul de la țintă. Ele dezvăluie punctele vulnerabile ale unei rețele. Exemple de scanere: **NSS** (Network Security Scanner - este un scanner destul de necunoscut, scris în Perl), **Strobe** (este un scanner care înregistrează toate porturile de pe o mașină dată), **Jakal** (este un scanner invizibil care poate scană un domeniu, în spatele unui firewall, fără a lăsa vreo urmă a sa).
- **Spărgătorul de parole** (password cracker), un program care poate decripta parole sau care poate dezactiva în vreun fel protecția prin parole. Un spărgător de parole nu are nevoie însă, să decripteze ceva. De obicei, funcțiile de criptare a parolelor sunt one-way, și deci parolile nu pot fi decriptate printr-o funcție inversă. Multe spărgătoare de parole nu sunt altceva decât programe care folosesc "forță brută", adică încearcă fiecare cuvânt (căutare exhaustivă). Exemple: **Crack** (unul din cele mai cunoscute), **Merlin** (program pentru administrarea spărgătoarelor de parole, a scannerelor și a altor utilitare de securitate).
- **Interceptorul**, dispozitiv, hardware sau software, care "capturează" informațiile transportate prin rețea.

Dispozitivele Software pot fi și destructive, ele sunt programe sau tehnici de hărțuire sau distrugere a datelor, exemple: troienii, virușii și instrumentele de refuz al serviciului. Acestea pot provoca pagube importante sau pot crea breșe în securitatea serverului.

Dispozitivul firewall este folosit pentru prevenirea accesului din exterior într-o rețea internă, este de obicei o combinație de hardware și software. De cele mai multe ori, dispozitivele firewall implementează scheme de excludere sau reguli care filtrează adresele dorite și nedorite.

Există multe tipuri de dispozitive firewall, fiecare având avantajele și dezavantajele lui. Cel mai întâlnit tip este firewall-ul la nivel de rețea este firewallul bazat pe ruter care este foarte rapid.

BIBLIOGRAFIE

1. Tanenbaum A., *Rețele de calculatoare*, ediția a patra, Byblos, București, 2003.
2. Popa M., *Bazele modelării rețelelor de calculatoare*, Editura Universității din București, 2004.
3. Wyatt A., L. *Succes cu INTERNET*; Editura și Atelierele Tipografice Metropol; București 1995.
4. Stallings W., *Local and Metropolitan Area Networks*, McMillian, 1993.
5. Zimmerman P., *An Introduction to Cryptography*, Network Asociates, 1998.
6. Klander L., *Anti Hacker – Ghidul securității rețelelor de calculatoare*, Editura All, București, 1998.
7. Peterson L L., Davie B.S. *Rețele de calculatoare*, Editura All, 2001.
8. Peterson L.L., Davie B.S. *Computer Network – A Systems Approach*, California, 1996.

PROGRAMARE PROCEDURALĂ

Lector univ. SILVIU BÂRZĂ

I. FORMATUL UNITĂȚILOR DE PROGRAM PASCAL

Limbajul PASCAL în formele sale clasice permitea realizarea unui singur tip de unitate de lucru numită unitate de program principal. Restricția care se aplică la o astfel de unitate de program se referă la memoria totală ocupată, atât de partea de proces de prelucrare cât și de partea de date la care procesul se aplică. Limitarea de memorie impusă de la prima versiune a limbajului și rămasă valabilă și în prezent este de 64 K# memorie (prin K# se notează un Kilo-octet de memorie, adică $2^{10}=1024$ octeți).

Odată cu apariția versiunii 5.0 a limbajului, s-a introdus și s-a dezvoltat un nou element de tipul unitate de program și o modalitate de realizare a programelor executabile pentru care teoretic nu se mai fac limitări de memorie. Noțiunea introdusă poartă numele de UNIT și reprezintă o colecție de date și de prelucrări, transformate în cod executabil și care se poate adăuga la programul principal pentru a forma un program executabil.

Mai mult, însăși forma de lucru pentru limbaj este dezvoltată prin această nouă manieră de lucru.

Versiunile cele mai recente ale limbajului PASCAL (7.0) au însemnat, din punctul de vedere al unităților de program, introducerea noțiunii de bibliotecă ca al treilea tip de unitate de program care se poate utiliza. Diferența dintre aceasta și noțiunea de unit constă în legarea de program la momentul execuției și, astfel, neinfluențând dimensiunea statică a programului executabil obținut dintr-o unitate de program principal.

Fiecare unitate de program are reguli stricte de formare și identificare care permit scrierea și utilizarea pentru obținerea unei aplicații. Formatele unităților vor fi descrise mai jos, fără a se indica și modalitățile de lucru care să permită exploatarea la maxim a lor, modalități care vor constitui teme pentru discuțiile ce vor urma în cadrul altor capitole.

În orice unitate de program, în orice linie și oriunde în linie se pot introduce comentarii. Un comentariu este orice sir de caractere cuprins într-o pereche de separatori speciali. O pereche posibilă de separatori pentru comentarii sunt accoladele, { la începutul și } la sfârșitul comentariului. Altă pereche de separatori este formată din secvențele de caractere (* și *) la începutul și, respectiv, sfârșitul comentariului.

I.1. Unitatea de program principal

Unitatea de program principal conține partea principală a unui program. Programul conține trei elemente distințe, și anume:

- un element de identificare al unității de program și care formează totdeauna prima linie a acesteia
- o zonă de definiții pentru elementele cu care se lucrează în program
- o zonă care conține instrucțiunile programului principal.

Antetul programului principal este format din cuvântul cheie PROGRAM și dintr-o identificare a programului dată printr-un identificator a cărui utilizare este interzisă în continuare. Acest identificator este recunoscut la nivel global și identifică programul din punct de vedere intern. Antetul se termină prin caracterul punct și virgulă.

Zona de definiții este formată din două subzone distințe, și anume

- subzona pentru definirea entităților de lucru din program. Aceste entități vor purta numele de entități globale, deoarece ele vor fi recunoscute pe parcursul întregului program principal, începând cu linia definirii lor, cu excepția unor redefiniri la nivelul subprogramelor cuprinse în programul principal.

Zona cuprinde definiții de tipuri, constante, variabile și eventuale etichete. Este bine ca utilizarea etichetelor să fie evitată cât mai mult posibil.

Pentru realizarea definirilor, se utilizează cuvintele cheie TYPE, CONST, VAR și LABEL. Fiecare definiție se încheie prin caracterul punct și virgulă.

- subzona de definire a subprogramelor interne programului principal. Subprogramele definire în acest loc poartă numele de subprograme globale, deoarece definirea lor este recunoscută pe parcursul întregului program imediat, după ce li se încheie definirea.

Ultima zonă a programului principal este formată din instrucțiunile programului principal. Zona poate fi considerată ca instrucțiune compusă urmată de un caracter punct.

Împărțirea pe zone a programului este prezentată cel mai bine prin schema 1.

O subzonă a fost împărțită doar pentru a semnala poziții obligatorii.

În această definire a structurii unui program, definiția USES din subzona de definire a entităților specifică introducerea elementelor definite într-o altă unitate de program, de tipul UNIT, elemente care se utilizează în programul principal curent. Dacă în programul principal realizat nu se utilizează nici un unit, atunci linia corespunzătoare definiției USES lipsește complet.

I.2. Unitatea publică de date și subprograme

Această unitate de program, cunoscută sub numele de UNIT, se utilizează pentru definirea, în exteriorul unor programe principale, a unor entități pentru date și a unor subprograme. De asemenea, un UNIT poate conține și o parte de cod executabil care este interpretat ca secvență ce se execută automat la începutul programului care utilizează UNIT-ul.

Elementele definite în cadrul unui UNIT și recunoscute în interiorul acestuia poartă numele de entități publice. Aceste entități sunt recunoscute atât în UNIT-ul în care sunt definite, cât și în orice altă unitate de program care conține numele UNIT-ului în care sunt definite în paragraful USES.

Paragraful USES, care am văzut că apare în zona de definiții, indică utilizarea, în unitatea de program în care apare, a unor entități definite în UNIT-uri. Formatul paragrafului este

USES *lista_unit*;

unde *lista_unit* este formată din identificarea unui *unit* sau dintr-o succesiune de identificări de UNIT-uri separate prin virgulă. Forma generală a unei identificări de unit este numele fișierului în care UNIT-ul se găsește sub forma sa compilată. Acest fișier are una din următoarele extensii:

- .TPU pentru UNIT-urile realizate sub limbajul PASCAL în formele executabile sub sistemul de operare DOS;
- .TPW pentru UNIT-urile realizate sub limbajul PASCAL în formatele executabile sub o versiune a sistemului de operare WINDOWS;
- .DCU pentru UNIT-urile realizate sub limbajul DELPHI și care se execută doar sub diversele versiuni ale sistemului de operare WINDOWS.

În partea de limbaj și auxiliare ale limbajului de programare, firma producătoare pune la dispoziție o serie de UNIT-uri de interes general și a căror conținut facilitează dezvoltarea mai rapidă a aplicațiilor. Dintre acestea, doar un singur UNIT are un regim special, în sensul că nu este indicat ca UNIT utilizat, și anume cel cu identificarea SYSTEM. În rest, utilizarea oricărui alt UNIT trebuie indicată prin prezența identificării sale în specificația USES.

La o compilare simplă, absența formei compilate pentru un UNIT utilizat produce eroare de compilare.

Pentru programele principale scrise sub forma DELPHI, identificarea unui UNIT se poate realiza prin:

- numele fișierului care conține forma compilată, și cu extensia .DCU. Absența acestei forme, la o compilare simplă, produce eroare;
- construcția formată din numele intern al UNIT-ului și identificarea fișierului în care UNIT-ul se găsește sub formă de UNIT sursă. Această construcție este de forma (*ident_unit* reprezentă identificarea internă a UNIT-ului iar *spec_unit* reprezentă specificația de fișier pentru fișierul care conține UNIT-ul sub formă sursă):

ident_unit IN ‘*spec_unit*’

PROGRAM <i>ident program</i> ;		zonă antet
USES <i>lista_unituri</i> ;	subzonă entități globale	zonă de definiții globale
TYPE : CONST : VAR : LABEL ...		
<i>Subprogram</i> ₁ ; <i>Subprogram</i> ₂ ; : <i>subprogram</i> _k ;	subzonă subprograme	
BEGIN		zonă de instrucțiuni
<i>Instrucțiune</i> ₁ ; <i>Instrucțiune</i> ₂ : <i>instrucțiune</i> _n		
END.		

Schemă 1. Formatul programului principal PASCAL

Un UNIT este format din următoarele zone:

- Antetul UNIT-ului;
- Zona de interfață prin care se declară elementele din UNIT la care se face acces din exteriorul acestuia (entitățile publice);
- Zona de implementare prin care sunt descrise procesele date sub formă de subprograme și la care se face acces din exterior și sunt definite elementele care se utilizează în acestea fără a putea fi utilizate din exterior, sau cele care permit inițializări ale variabilelor accesate din exterior. Elementele definite în această zonă poartă numele de entități private, deoarece ele sunt recunoscute după definire doar în interiorul UNIT-ului;
- Zona de instrucțiuni executabile prin care se realizează inițializări.

Antetul unui UNIT determină un nume al unitului, nume considerat tot ca entitate publică. Numele dat ca identificare internă pentru UNIT trebuie să coincidă cu numele fișierului extern în care sursa UNIT-ului va fi memorată. Acest fișier va avea extensia .PAS. Astfel, pentru formatul general al antetului

UNIT ***ident_unit***;

este necesară memorarea sursei UNIT-ului în fișierul cu identificarea ***ident_unit.PAS***. Din această cauză, ***ident_unit*** trebuie să fie nu numai un identificator PASCAL corect, ci și un nume corect pentru o specificație de fișier din punctul de vedere al sistemului de operare.

Zona de interfață a unui UNIT conține doar declarații de entități, separate în subzone distințe și anume:

- antetul zonei care constă în cuvântul cheie INTERFACE;
- subzona de specificare a UNIT-urilor ce sunt utilizate în UNIT-ul curent și a căror elemente devin publice și în cazul în care la utilizarea acestuia nu se indică și UNIT-urile implicate de el; dacă în UNIT-ul realizat curent nu se fac referiri la alte UNIT-uri, atunci această subzonă lipsește complet;
- subzona de definire a entităților relative la datele ce vor fi utilizate ca publice; dacă în UNIT-ul curent nu se definesc entități relative la date atunci subzona lipsește complet;
- subzona de definire a proceselor date sub formă de subprograme publice; definirea subprogramelor se realizează prin liniile lor complete de identificare; dacă UNIT-ul nu conține procese descrise sub formă de subprograme, atunci subzona lipsește complet.

Zona de implementare conține, la rândul ei, subzone distințe, și anume:

- antetul zonei format din cuvântul cheie IMPLEMENTATION;
- subzona de specificare a UNIT-urilor considerate private, care vor fi utilizate în UNIT-ul curent; dacă în UNIT-ul curent nu se folosește în mod privat nici un alt UNIT, atunci subzona lipsește complet;
- subzona de definire a entităților private relative la date; dacă nu se folosesc date private, atunci subzona lipsește complet;
- subzona de declarare a subprogramelor private utilizate în cadrul UNIT-ului curent, dacă există astfel de subprograme; în caz contrar, subzona este în întregime absentă;
- subzona de declarare a subprogramelor care au fost deja definite ca publice în zona de interfață; dacă în zona de interfață lipsește subzona de definire a subprogramelor publice, atunci și subzona de declarare a lor este absentă.

UNIT <i>ident unit</i>;		zonă antet
INTERFACE	subzonă antet interfață	zonă de interfață
USES <i>unituri_publice</i> ;	subzonă utilizare UNIT-uri publice	
TYPE ⋮ CONST ⋮ VAR ⋮	subzonă definire date publice	
<i>Antet_subprogram_public₁</i> ; <i>antet_subprogram_public₂</i> ; ⋮ <i>antet_subprogram_public_k</i> ;	subzonă definire subprograme publice	
IMPLEMENTATION	subzonă antet implementare	zonă de implementare
USES <i>unituri_private</i> ;	subzonă utilizare UNIT privat	
TYPE ⋮ CONST ⋮ VAR ⋮ LABEL ...	subzona date private	
<i>Subprogram_privat₁</i> ; <i>subprogram_privat₂</i> ; ⋮ <i>subprogram_privat_m</i>	subzona declarare subprograme private	
<i>Subprogram_public₁</i> ; <i>Subprogram_public₂</i> ; ⋮ <i>subprogram_public_k</i>	subzona declarare subprograme publice	
BEGIN <i>Instrucțiune₁</i> ; <i>Instrucțiune₂</i> ; ⋮ <i>instrucțiune_n</i>	subzonă instrucțiuni	zonă de instrucțiuni direct executabile
END.	subzonă terminare UNIT	

Schema 2. Formatul UNIT-urilor PASCAL

Zona de instrucțiuni din UNIT este formată din:

- subzona de instrucțiuni prin care se specifică aşa-numite procese de inițializare prin UNIT și conține instrucțiuni valide în limbajul PASCAL; în cazul în care UNIT-ul nu conține secvențe de inițializare, subzona lipsește complet; în cazul în care subzona este prezentă, ea începe prin cuvântul cheie BEGIN;
- subzona de indicare a terminării UNIT-ului formată din cuvântul cheie END, urmat de caracterul punct.

Prezentarea sumară a unui UNIT este descrisă cel mai bine prin schema 2.

I.3. Unitatea de execuție dinamică a subproceselor

Această unitate de program PASCAL a apărut la trecerea realizării și execuției programelor pentru sistemele de operare din clasa WINDOWS. Prin utilizarea acestor unități de program se obțin fișiere cu instrucțiuni în format executabil și care sunt legate efectiv de program în mod dinamic, la execuția acestuia de unde și numele sub care sunt cunoscute formele lor compilate, biblioteci cu legare dinamică (DLL, prescurtare din engleză de la Dynamic-Link Library).

Utilizarea unui astfel de mod de lucru are avantajul posibilității de a lucra cu programe care în faza de compilare au dimensiuni relativ mici, iar în execuție au o dimensiune variabilă, în memoria utilizată fiind prezente doar componenta statică (programul principal) și acele componente de tip DLL care sunt efectiv utilizate.

Suplimentar, dacă două programe aflate în execuție folosesc simultan părți ale aceluiași DLL, acesta este încărcat o singură dată și utilizat în comun (spunem că are loc o partajare a procesului între două sau mai multe programe).

Operația de trecere a subprogramelor din DLL la programul principal în care sunt apelate se numește export, iar pentru subprogramele din DLL care pot fi utilizate din exterior această specificație este obligatorie.

O problemă o constituie faptul că între un apelator și un DLL transferul de date se face doar la nivelul transferurilor subprogram și nu direct, cum este cazul pentru UNIT-uri.

Formatul general pentru o unitate de execuție dinamică este similară cu unitatea de program principal. Un element de diferențiere îl reprezintă antetul unității de program care folosește cuvântul cheie LIBRARY în locul cuvântului cheie PROGRAM. Un aspect deosebit este constituit din definirea proceselor de comunicare cu exteriorul, ca procese exportabile. Prezentarea aspectelor efective legate de realizarea acestor unități de program se va face într-un capitol ulterior legat doar de problematica DLL-urilor.

II. SCRIEREA SUBPROGRAMELOR ÎN UNIT-URI

Pentru realizarea subprogramelor publice în cadrul UNIT-urilor se poate utiliza același procedeu de realizare ca pentru scrierea în unitatea de program în care este apelat. Acest mod de scriere, aşa cum am văzut, impune definirea în unitatea respectivă de program, înaintea utilizării, a tipurilor utilizator corespunzătoare tipurilor structurate. Această cerință duce la definirea tipului utilizator ca tip public, și astfel, obligativitatea de a folosi doar acest tip pentru definirea variabilelor care se folosesc la apel, pentru parametrii corespunzători.

Lucrul sub forma descrisă produce o limitare destul de mare la utilizarea subprogramelor publice și deci, și o limitare corespunzătoare a utilizării întregului UNIT. Subiectul acestui subcapitol este realizarea unor subprograme, încât limitările utilizării subprogramelor să se reducă cât mai mult posibil la utilizarea datelor structurate predefinite (masive și siruri de caractere cu lungime fizică explicită).

Realizarea subprogramelor va utiliza o clauză a definirii variabilelor prin care același spațiu de memorie se poate utiliza pentru structuri diferite, operație cunoscută ca partajarea spațiului de memorie între date. Modalitățile efective de realizare a subprogramelor vor fi descrise apoi pentru vectori și matrici, pentru sirurile de caractere realizarea fiind similară realizării pentru vectori.

Partajarea datelor este un instrument util pentru economisirea memoriei utilizate static în cadrul programului pentru date structurate de dimensiuni mari în cazurile în care prelucrările unor astfel de structuri se face distinct în procesul realizat.

II.1. Partajarea memoriei pentru date

Fenomenul de partajare a memoriei pentru date este o tehnică de inginerie a programării și are la bază utilizarea adreselor din memorie pentru regăsirea datelor. Fiecare identificator de date utilizat la nivelul unui program este asociat cu o adresă de memorie, care indică începutul informației utile asociată identificatorului. Așa cum am văzut în capitolul relativ la tipurile de date, pentru fiecare tip se folosește un număr de octeți, care se poate determina prin relații relative la tipul variabilei.

Partajarea memoriei se realizează prin considerarea a doi sau mai mulți identificatori care au asociată aceeași adresă de memorie. Mai mult, pentru o căt mai mare libertate de lucru, nu există restricții de tip pentru o pereche de identificatori de variabilă care au asociată aceeași adresă.

Partajarea memoriei se realizează necondiționat, la momentul definirii variabilelor, printr-o construcție:

ident_nou:tip_variabilă ABSOLUTE ident_vechi;

unde ***ident_nou*** este identificarea definită curent, ***tip_variabilă*** este tipul pentru variabila definită curent, iar ***ident_vechi*** reprezintă una din variabilele deja definite a cărei adresă de început va fi și adresa de început a variabilei definite curent.

II.2. Subprograme cu partajarea memoriei pentru datele de tip masiv unidimensional

Realizarea subprogramelor poate utiliza tehnica partajării memoriei pentru datele de tip masiv unidimensional, aplicată între un parametru și o variabilă definită local. Pentru realizarea independenței relative la definirea parametrului acesta se indică în lista de parametri printr-o construcție

VAR *ident_parametru*

Prezența acestei definiții specifică un parametru transmis prin adresă și pentru care nu se indică tipul. Un avantaj suplimentar este că nu mai trebuie definit un tip utilizator și, astfel, variabila specificată în poziția acestui parametru poate fi definită oricum în programul în care se realizează apelul.

Specificarea tipului care se va utiliza în subprogram este realizată la nivelul unei variabile locale, care va partaja memoria cu specificația pentru parametru. Astfel, singura restricție care se impune este cea relativă la tipul elementului variabil, utilizat în poziția parametrului. Prelucrarea din subprogram va utiliza definiția locală și nu cea pentru parametru (care ar produce eroare prin absența tipului).

Utilizarea acestei tehnici în absența verificării domeniilor nu creează nici un fel de probleme în execuția programelor. Pentru opțiunea verificării domeniilor activate, există două variante de lucru în realizarea subprogramelor.

O primă variantă este de restricționare suplimentară la o dimensiune maximală, egală cu cea dată la specificarea tipului pentru variabila locală corespondentă parametrului.

A doua posibilitatea este utilizarea directivei de compilare corespunzătoare verificării domeniilor prin care verificarea să fie inhibată la începutul subprogramului și reactivată la terminarea acestuia. În acest mod, numărul de componente cu care se definește masivul local nu mai influențează realizarea prelucrării din subprogram.

Realizarea subprogramelor folosind tehnica partajării memoriei se face cu preponderență în cadrul UNIT-urilor (pentru subprogramele publice) și DLL-urilor (pentru subprogramele de comunicare), dar se poate aplica cu succes și la realizarea subprogramelor locale sau în cadrul programului principal, aşa cum se va vedea în următorul exemplu, reluarea exemplului 11.4.-2 în ceea ce privește realizarea subprogramului.

Modificăm însă programul principal, pentru a permite realizarea operațiilor pentru doi vectori definiți diferit, prin care se prezintă mai bine utilitatea modului de construcție. Realizăm această reluare a exemplului pentru a se putea vedea diferențele care intervin între cele două modalități de realizare a subprogramelor. În plus vom utiliza analiza parametrilor realizată în exemplul menționat.

II.3. Subprograme cu partajarea memoriei pentru datele de tip masiv bidimensional

Pentru realizarea subprogramelor care folosesc schema de partajare a memoriei asupra masivelor bidimensionale, principiul utilizat este similar celui prezentat în paragraful anterior, deoarece, în acest caz, masivele sunt considerate tot unidimensionale, ceea ce corespunde modalității interne de memorare.

Reținerea datelor sub forma de masiv bidimensional se realizează prin locații succesive de memori, la fel ca pentru vectori. Determinarea poziției în această succesiune de valori se face printr-o formulă numită formulă de rang.

Pentru masivele multidimensionale oarecare, posibil de definit ca structuri de date PASCAL, formula de rang aplicată este o funcție dependentă de numărul dimensiunilor (domeniilor) specificate și de forma pe care domeniile o au. Formula de rang intern pentru determinarea unei valori dintr-un masiv depinde și de numărul de octeți utilizați pentru tipul de bază și reprezintă deplasarea care se aplică la adresa de început a variabilei pentru a obține adresa elementului dorit.

Formula pe care o considerăm în realizarea partajării memoriei consideră că domeniile de valori sunt întregi și că prima valoare a domeniului este întotdeauna 1. De asemenea, adresa abținută reprezintă valoarea pentru indicele dintr-un vector cu același tip de bază și care s-ar memora în același spațiu cu masivul bidimensional. Corespondența dată este o bijecție dată pe perechile de indicii pentru identificarea elementului din matrice cu valoarea pe nulțimea indicilor de vectori corespunzători, deci

$rang: \{1, 2, \dots, n\} \times \{1, 2, \dots, m\} \rightarrow \{1, 2, \dots, nm\}$, dată prin $\text{rang}(i, j) = (i-1)m + j$. Valorile n și m reprezintă pentru matricile definite în cadrul programului (static) limitele superioare ale domeniilor din definire, și nu dimensiunile efective.

Realizarea partajării memoriei impune și cunoșterea dimensiunii de definiție pentru numărul de coloane (mărimea superioară pentru a doua dimensiune) pentru matricea (masivul bidimensional) din programul care face apelul. Astfel, pentru fiecare matrice considerată parametru pentru subprogram va trebui să introducem suplimentar un parametru pentru acest număr maximal de coloane. Este de înțeles că, pentru lucrul cu mai multe matrici dimensionate identic, se va introduce un singur parametru pentru număr maxim de coloane (acest lucru însă reduce generalitatea aplicării).

Facem doar observația că pentru parametrul de dimensionare se indică transmiterea prin valoare, pentru a permite ca la apel să se poată utiliza și valori constante.

Rezolvarea este similară rezolvării pentru masive unidimensionale, doar că, în locul indicelui pentru element, se va aplica funcția de rang pe care am indicat-o mai sus.

III. RECURSIVITATEA ÎN LIMBAJUL PASCAL

Rezolvarea normală a problemelor se face în general pe baza unor procese iterative. Există însă și probleme a căror rezolvare, aşa cum apare și teoretic, nu se poate rezolva decât prin procese care se autoapeleză, și astfel se numesc recursive.

Pentru înțelegerea fenomenului de recursivitate este necesară definirea unor elemente importante cu legătură directă cu realizarea programelor.

Definiție. Dacă A este mulțimea subprogramelor utilizate, la care se adaugă programul principal pp , și considerăm mulțimea V a perechilor (a,b) , pentru care din a (program principal sau subprogram) se face un apel la b (subprogram), atunci spunem că $G=(A,V)$ definește graful de apel al programului principal pp .

După cum se vede din definiția dată, graful de apel pentru un program este un graf orientat și care conține un nod cu gradul de intrare 0 (care este chiar programul principal).

Definiție. Fie $G=(A,V)$ un graf orientat cu A mulțimea de noduri și V mulțimea arcelor ($V \subset A \times A$). Spunem că graful G conține bucle (sau 1-circuite, circuite de lungime 1) dacă $(\exists)x, x \in A$, astfel încât $(x,x) \in V$.

Definiție. Fie $G=(A,V)$ un graf orientat cu A mulțimea de noduri și V mulțimea arcelor ($V \subset A \times A$). Spunem că graful G conține k -circuite (sau circuite de lungime k , $k \geq 1$) dacă $(\exists)v_1, v_2, \dots, v_k$, $v_i = (x_i, y_i) \in V$, $(\forall)i=1,2,\dots,k$, astfel încât $v_i \neq v_j$ $(\forall)i \neq j$, $y_i = x_{i+1}$ $(\forall)i=1,2,\dots,n-1$ și $x_1 = y_k$.

Folosind noțiunea de graf de apel, putem da definiții de caracterizare a programelor din punctul de vedere al modalității de rezolvare a problemelor.

Definiție. Spunem că un program este **iterativ** dacă graful său de apel nu conține circuite.

Spunem că un program este **recursiv de prima speță**, dacă graful de apel al programului conține cel puțin o buclă, dar nu conține circuite cu lungimea mai mare sau egală cu 2.

Spunem că un program este **recursiv de speță a doua**, dacă graful de apel al programului conține cel puțin un circuit de lungime mai mare sau egală cu 2.

Recursivitatea este permisă în limbajul PASCAL, prin specificare directă. Compilatorul recunoaște ambele tipuri de recursivitate și generează instrucțiuni mașină corespunzătoare, care permit rezolvarea recursivității. Aceste instrucțiuni realizează o salvare a stării programului în momentul fiecărei intervenții a unui apel care implică recursivitate.

Starea programului la apelarea unui subprogram poartă numele de context de apel și este dat prin definiția:

Definiție. *Contextul de apel* care intervene în limbajul PASCAL la cererea de execuție a unui subprogram cu o formă de recursivitate este format din următoarele elemente:

- adresa de revenire după execuția completă a instrucțiunilor subprogramului;
- totalitatea valorilor pentru parametrii transmiși prin valoare;
- totalitatea valorilor variabilelor locale din momentul apelului;
- elementul de returnare a rezultatului, doar dacă subprogramul recursiv este de tipul FUNCTION.

Codului executabil generat care permite rezolvarea recursivă realizează salvarea contextului de apel în stiva program înainte de a trece la prima instrucțiune executabilă a subprogramului, ca, apoi, la terminarea execuției integrale a subprogramului, să restaureze acest context de apel și, apoi, să treacă la instrucțiunea care urmează apelului.

III.1. Forma generală a subprogramelor recursive

Realizarea proceselor recursive impune, în primul rând, definirea rezolvării matematice a problemei la nivel de relație de recurență. O astfel de definiție se realizează printr-o relație generală de calcul, prin care, pentru un pas de calcul se utilizează rezultatele din pașii anteriori și printr-o relație inițială pentru realizarea calculului.

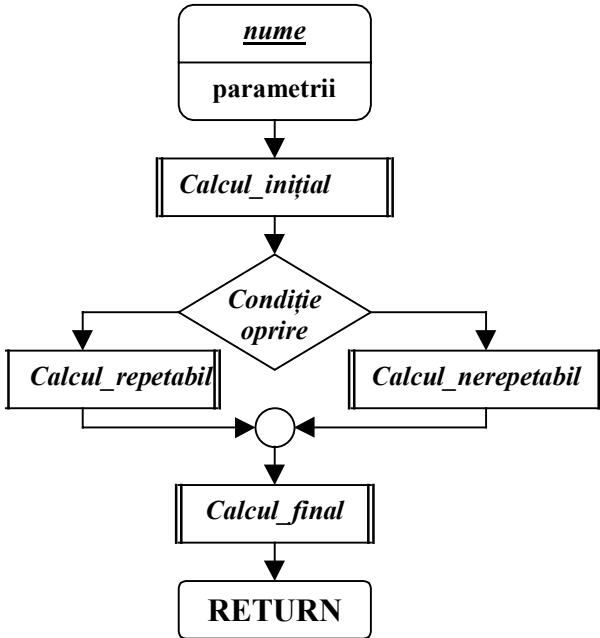
Obținerea unui subprogram recursiv trebuie să utilizeze elementele matematice. Prelucrarea se realizează în funcție de pasul curent, fiind necesară diferențierea, prin structură alternativă, a aplicării relației generale sau a celei inițiale. Astfel, efectuarea calculelor se va face printr-o structură alternativă, și nu prin una repetitivă ca în cazul prelucrărilor iterative.

Condiția prin care se realizează diferențierea între aplicările diverselor formule poartă numele de **condiție de oprire a recursivității**.

La nivelul prezentării algoritmilor, pentru rezolvările recursive se respectă următoarele schematici:

- pentru reprezentarea PSEUDOCOD
- proces nume_proces(parametrii)
- BEGIN
- Calcul_inicial*
IF *Condiție_oprire*
THEN *Calcul_nerepetabil*
ELSE *Calcul_repetabil*
FI
Calcul_final
- END

- pentru reprezentarea prin schemă logică



În prezentările de mai sus:

- **Calcul_inicial** reprezintă un proces de prelucrare care se aplică întotdeauna anterior aplicării formulelor recurente care conduc la recursivitate. Acest proces poate fi absent. El este obligatoriu atunci când condiția care intervine nu este una directă asupra parametrilor ci implică la rândul ei un proces de calcul;
- **Condiție_oprire** reprezintă o expresie logică prin care se selectează aplicarea relației inițiale sau celei generale din definirea rezolvării recurente. Ea poartă numele de condiție de oprire referitor la fenomenul recursiv, deoarece determină întreruperea aplicării formulei generare;
- **Calcul_nerepetabil** indică prelucrările realizate pentru utilizarea în calcul a relației inițiale din definirea recurrentă. Acest proces poate fi vid;
- **Calcul_repetabil** reprezintă partea de prelucrare corespunzătoare relației generale din rezolvarea recurrentă și este singura parte din subprogram în care se ajunge la apariția unui circuit în graful de apel al programului care apelează această subprogram;
- **Calcul_final** se utilizează pentru a indica realizarea de prelucrări indiferent de aplicarea în recurrentă a relației inițiale sau a celei generale. Acest bloc poate fi absent.

III.2. Rezolvarea recursivității de prima spătă

Din punct de vedere al realizării unui subprogram în limbajul PASCAL, nu există nici o diferență față de subprogramele iterative. Un subprogram recursiv este recunoscut prin fenomenul de autoapelare. O problemă care poate interveni la prelucrările de tip recursiv este generată de eventualul număr mare de apeluri care conduc la epuizarea stivei program. Acest lucru se poate rezolva prin modificarea dimensiunii stivei sistem prin opțiunile meniului sau prin utilizarea directivei de compilare \$M.

Dacă subprogramul este de tip PROCEDURE, atunci între instrucțiunile subprogramului se află cel puțin o instrucțiune de apel de procedură, în care apelul se face la procedura în curs de scriere.

Dacă subprogramul este de tip FUNCTION, atunci subprogramul apare în instrucțiunile sale cel puțin o dată într-o expresie.

În absența recursivității, pentru apelul unui subprogram se realizează completarea stivei cu adresa de revenire la apelator, valorile parametrilor transmiși prin valoare, adresele de început ale parametrilor transmiși prin adresă și, doar în cazul funcțiilor, se rezervă memorie conform tipului funcției pentru returnarea rezultatului. La revenirea din subprogram se elimină din stivă toate elementele depuse cu excepția locației pentru returnarea rezultatului pentru subprogramele funcție.

Prezența recursivității impune ca, pe lângă elementele enumerate mai sus, în stivă să fie trecute și valorile pentru variabilele locale utilizate în subprogram. La revenirea dintr-un apel recursiv, înainte de eliminarea din stivă are loc și redepunerea valorilor variabilelor locale, din stivă în locul de proveniență.

Prelucrarea datorată apariției recursivității produce depunerile repetitive în stiva program până la întâlnirea îndeplinirii condiției de oprire a recursivității. Această depunere succesivă poartă numele de expandarea recursivității pe stiva program și spunem că operația are sens crescător datorită creșterii mărimii stivei. După întâlnirea îndeplinirii condiției de oprire a recursivității se începe realizarea efectivă a calculelor prin aplicarea procesului de calcul pentru relația inițială și apoi a relațiilor generale, fiecare aplicare de relație fiind urmată din reveniri cu diminuarea stivei. Partea a doua a rezolvării recursive poartă numele de calcul în sens descrescător.

Exemplul care urmează va prezenta atât realizarea unui subprogram recursiv și apelarea sa, cât și modalitatea completă de rezolvare a recursivității. În stiva program se vor trece, pentru exemplificare, doar elementele legate strict de rezolvarea recursivă.

Exemplu. Subprogramul realizat determină valoarea pentru combinări de n luate câte k (care în matematică are notația C_n^k sau $\binom{n}{k}$). Această valoare are valoarea inițială $C_n^0 = 1$ pentru orice număr natural n și are relația de recurență

$C_n^k = \frac{n}{k} C_{n-1}^{k-1}$. Subprogramul realizat va fi de tip FUNCTION și va avea ca parametri două valori de tip BYTE. Datorită valorilor mari care se obțin în urma realizării calculelor, rezultatul va fi dat ca număr real, chiar dacă matematic se obține o valoare naturală. Programul nu va verifica corectitudinea valorilor pentru care se calculează și presupune că valorile date îndeplinesc condiția matematică $0 \leq k \leq n$. Valoarea reprezintă numărul submulțimilor cu k elemente dintr-o mulțime cu n elemente.

Exemplul ales este mult simplificat, deoarece subprogramul utilizat nu conține variabile locale și, astfel, în stivă, se vor afla doar valorile parametrilor (nu avem nici parametrii transmiși prin adresă) și locația pentru returnarea rezultatului.

Programul PASCAL realizat este:

```
PROGRAM calcul_comb_n_k;
VAR
  n,k:BYTE;
  v:REAL;
FUNCTION Combin(n,k:BYTE) : REAL;
BEGIN
  IF k=0 THEN
    Combin:=1.0
  ELSE
    Combin:=n*Combin(n-1,k-1)/k
  END;
BEGIN
  WRITE('Numar total elemente:');
  READLN(n);
  WRITE('Numar elemente submultime:');
  READLN(k);
  v:=combin(n,k);
  WRITELN('Numar submultimi:',v:15:0)
END;
```

Pentru exemplificarea modului de execuție, vom considera că s-au introdus valorile 7 și 3 pentru n și respectiv k . Primul apel intervine în programul principal pentru valorile 7 și 3, iar stiva va avea configurația arătată în figura 1.a. Deoarece valoarea pentru al doilea parametru este diferită de 0, condiția din subprogram nu este îndeplinită și se realizează un nou apel pentru valorile 6 și 2; stiva va acum completață conform figurii 1.b. Aceast apel produce, la rândul lui, apelul pentru valorile 5 și 1, care conduce la o stivă de forma prezentată în figura 1.c. Procesul de expandare continuă cu apelul pentru parametrii cu valorile 4 și 0 cu stiva program formată ca în figura 1.d. Acum, al doilea parametru are valoarea zero și urmează a se realiza atribuirea $\text{Combin}:=1$.

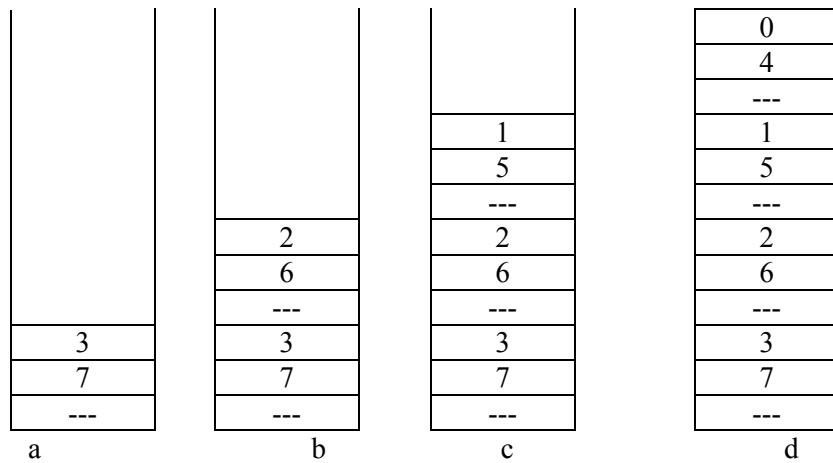


Figura 1. Evoluția stivei program la execuția exemplului pentru $n=7$ și $k=3$ (expandare recursivitate)

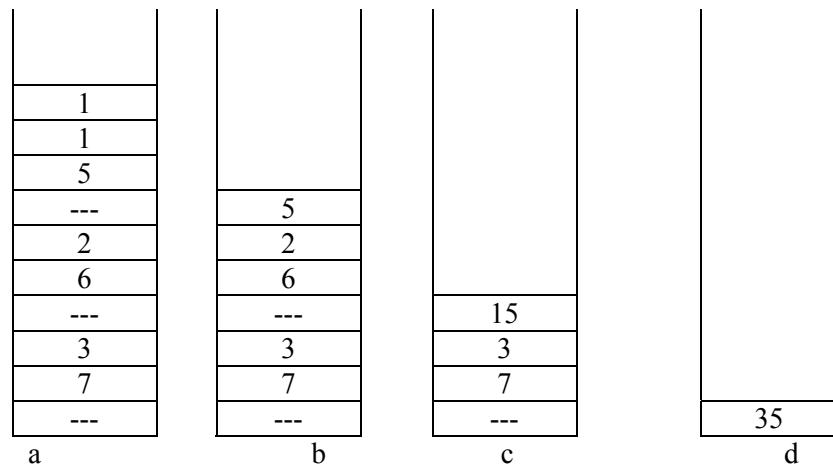


Figura 2. Evoluția stivei program la execuția exemplului pentru $n=7$ și $k=3$ (rezolvare calcul recursiv)

După prima aplicare de relație și prima revenire, valoarea pentru rezultatul funcției este în vârful stivei care are conținutul conform figurii 2.a. Urmează realizarea calculului efectiv pentru parametrii cu valorile 5 și 1 și revenirea la apelul anterior cu vârful stivei cu valoarea 5, aşa cum se vede în figura 2.b. Acum se realizează calculul cu această valoare și cu parametrii cu valorile 6 și 2, iar rezultatul, 15, este după revenire în vârful stivei, ca în figura 2.c. În sfârșit, are loc calculul cu valoarea 15 și parametrii cu valorile 7 și 3, iar revenirea se face, de această dată, la programul principal și valoarea din stiva din figura 2.d. reprezintă rezultatul final al calculului.

III.3. Rezolvarea recursivității de speță a două

Pentru rezolvarea la nivelul programului PASCAL al acestui tip de recursivitate s-a introdus o proprietate specială pentru definirea subprogramelor, și anume, definirea anticipată.

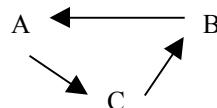
Definirea unui subprogram ca definiție anticipată se realizează prin aplicarea caracteristicii date prin cuvântul cheie FORWARD aplicat la antetul de subprogram. Pentru procedură, separarea listei de parametri de caracteristică se face prin caracterul punct și virgulă. Tot acest caracter se folosește și pentru separarea caracteristicii de caracteristica de tip rezultat pentru subprogramele FUNCTION. Declararea subprogramului ca definiție anticipată face ca restul subprogramului să nu mai fie definit în locul respectiv.

Introducerea declarării anticipate a fost făcută pentru a respecta regula de bază a scrierii programelor PASCAL, și anume, obligativitatea ca, la utilizarea unei entități, aceasta să fi fost definită anterior.

Conform definiției, recursivitatea de speță a două intervine când, într-un lanț de subprograme, unul din subprogramele apelate curent este reapelat. Problema principală pentru definirea rezolvării recursive la nivelul programului o constituie determinarea ordinii de definire a subprogramelor în cadrul unității de program.

Pentru a determina acest lucru, va trebui considerat circuitul din graful de apel. Se poate alege orice nod (subprogram) din acest circuit și pentru el să se realizeze definirea anticipată. Restul subprogramelor se definesc integral în ordinea inversă a parcurgerii arcelor circuitului.

Exemplu. Să considerăm un proces de calcul recursiv realizat prin intermediul unei funcții A și a două proceduri B și C, care formează, în graful de apel, circuitul:



Să considerăm procedura b ca procedură aleasă pentru declarație anticipată. În acest mod, la nivelul definirii se întrerupe circularitatea. Prin această alegere, ordinea de definire a subprogramelor va fi dată prin secvența:

```
PROGRAM exemplu;
:
PROCEDURE B; FORWARD;
PROCEDURE C;
:
BEGIN
:
END;
```

```
FUNCTION A:tip;  
:  
BEGIN  
:  
END;  
PROCEDURE B;  
:  
BEGIN  
:  
END;  
:
```

Așa cum se poate observa, definirile efective sunt în ordinea inversă a parcurgerii circuitului începând cu **B**.

BIBLIOGRAFIE

1. Albeanu G., *Algoritmi și limbaje de programare*, Editura Fundației România de Mâine, București, 2000.
2. Albeanu G., Radu L., *Algoritmica și programare în Pascal*, Editura Fundației România de Mâine, București, 2001.
3. Knuth D., *Arta programării calculatoarelor*, vol. I, *Algoritmi fundamentali*, Editura Teora, București, 2000.
4. Livovschi L., Georgescu H., *Analiza și sinteza algoritmilor*, Editura Științifică și Enciclopedică, București, 1974.
5. Albeanu G., *Tehnici de programare, Lucrări practice de programarea calculatoarelor*, Editura Fundația România de Mâine, București, 2003.
6. Bârză S., *Culegere de probleme de algoritmică și programare*, vol I, *Programare statică*, Editura Universității București, 2001.
7. Bârză S., *Algoritmica și programare. Note de curs*, vol. I, *Programare statică*, Editura Universității București, 2001.

BIROTICA

Lector univ. SILVIU BÂRZĂ

I. PLATFORMA WINDOWS

Platforma WINDOWS este o componentă importantă pentru desfășurarea activității pe sistemele de calcul de tipul IBM sau compatibile și care are rolul de a ușura activitatea de operare. Platformele de tipul WINDOWS au fost introduse pe calculatoarele din gama PC-urilor în anii '80 și sunt specifice utilizării mouse-ului ca echipament de intrare. Platformele au evoluat odată cu dezvoltarea calculatoarelor și utilizează facilitățile grafice existente hardware.

Activitatea se desfășoară prin intermediul ferestrelor specifice, care permit selectarea operațiilor, atât prin folosirea tastaturii, cât și prin intermediul cursorului mouse și acționarea butoanelor acestuia.

I.1. Fereastra DESKTOP

Ecranul calculatorului conține o fereastră generală, activă permanent, numită DESKTOP. Această fereastră este formată dintr-o zonă de selecție generală și o bară de activitate curentă sau de START.

Bara de activitate curentă se găsește, în general, în finalul ecranului și conține două elemente principale, butonul de START și zonele de interes general. În plus, această bară conține câte un buton pentru fiecare din aplicațiile lansate în execuție în mod curent și prin care aceste aplicații pot fi activate. La un moment dat, în afară de aplicația de ceas, doar o singură aplicație poate fi activă, cu excepția cazului în care se execută programe speciale, cum ar fi cele de imprimare și unele produse de multimedia. Acest lucru se explică prin tehniciile de multitasking, prin care procesorul preia, pe rând, pentru un timp scurt, sarcinile de prelucrare ale programelor inițiate.

Zonele de interes general cuprind aplicații considerate rezidente, fără a fi active, și care pot fi activate direct din bara de start. În zona de interes general este plasat și ceasul prin care se afișează continuu ora curentă a sistemului.

Zona de selecție generală conține elemente de indicare a aplicațiilor sau elementelor din sistemul de calcul la care se poate face acces imediat. Acestea sunt formate dintr-o imagine grafică de identificare, numită iconă (ICON) și din denumirea specifică. O pereche formată dintr-o iconă și denumirea specifică poartă numele de SHORTCUT.

Pentru accesul la o aplicație din platforma WINDOWS, pentru care există SHORTCUT, avem două variante, și anume, fie plasarea cursorului mouse pe SHORTCUT și acționarea dublă a butonului din stânga al mouse-ului (double click), fie utilizarea butonului de START.

Lansarea aplicațiilor utilizând butonul de START

Pentru a iniția execuția unei aplicații, indiferent dacă există sau nu și alte moduri de activare, se utilizează butonul de START, aflat în stânga barei de START. Prin acționarea cu click pe acest buton se deschide o bară de selecție cu o serie de elemente prin care se poate alege activitatea principală care urmează a fi activată. Pentru toate

calculatoarele, elementele prezente sunt: PROGRAMS, FAVORITES, DOCUMENTS, SETTINGS, FIND, HELP, RUN și SHUTDOWN.

ACTIONAREA cu click pentru *PROGRAMS* deschide o bară de selecție care conține principalele aplicații executabile sub platforma *WINDOWS*. Unele din aceste aplicații pot fi lansate direct, altele sunt formate din mai multe programe activabile separat și pentru acestea *actionarea* produce o nouă detaliere printr-o bară verticală de meniu.

SELECTIA RUN permite alegerea directă a unui fișier care urmează a fi executat și care poate fi chiar un program neconstruit pentru execuție sub platforma *WINDOWS*. La alegerea elementului de execuție se deschide o fereastră de dialog, care are bara de titlu formată din titlul *RUN* și butoanele de solicitare de asistență (butonul ?), de închidere a ferestrei și terminare a activității (butonul X). În partea inferioară a ferestrei, se dispune de trei butoane, *OK* (pentru închiderea ferestrei, terminarea activității și trecerea la deschiderea fișierului specificat), *CANCEL* (pentru închiderea ferestrei și terminarea activității) și *BROWSE* (pentru facilitarea găsirii fișierului dorit printr-un instrument de regăsire și selectare directă).



Figura 1.1. Fereastra de solicitare de execuție directă (RUN)

Fișierul la care se face acces trebuie dat printr-o specificație de fișier completă (cale, nume și extensie). Casetă de deschidere (OPEN) din fereastra de *dialog* permite specificarea numelui unui program, indicarea unui director, document sau resursă Internet care poate fi deschisă. Casetă este de tipul casetă cu selecție, lista de selecții conținând, de fapt, o listă de istoric al inițiierilor de execuții prin selecția *RUN*.

Actionarea cu click asupra butonului *BROWSE* produce deschide-rea unei noi ferestre de dialog, cu bara de titlu formată din titlul *BROWSE* și butoanele ? și X. Partea inferioară a acestei ferestre conține butoanele *OPEN* (selectează fișierul specificat în caseta FILE NAME, închide fereastra de dialog *BROWSE*, și se revine la fereastra *RUN* cu fișierul selectat de pe caseta OPEN) și *CANCEL* (închide dialogul *BROWSE* și revine la fereastra *RUN* neschimbă).

Tot în partea inferioară a ferestrei *BROWSE* se găsește caseta de introducere directă a specificației de fișier aleasă (FILE NAME) și caseta de selecție FILES OF TYPE care permite specificarea tipului fișierelor care se afișează în afara directoarelor. Se poate selecta între tipul PROGRAMS (fișiere de tip program) și tipul ALL FILES (toate fișierele).

Partea centrală a dialogului *BROWSE* este formată dintr-o casetă de listare, în care sunt afișate subdirectoarele directorului curent și fișierele din directorul curent, conform alegерii existente în caseta FILES OF TYPE. Intrarea într-un subdirector se face prin doubleclick pe numele directorului listat. Doubleclick pe un fișier din caseta de listare are ca efect selectarea fișierului și realizarea efectului butonului *OPEN*. Un click pe un fișier din caseta de listare produce selectarea fișierului a cărui specificație apare și în caseta de introducere FILE NAME.

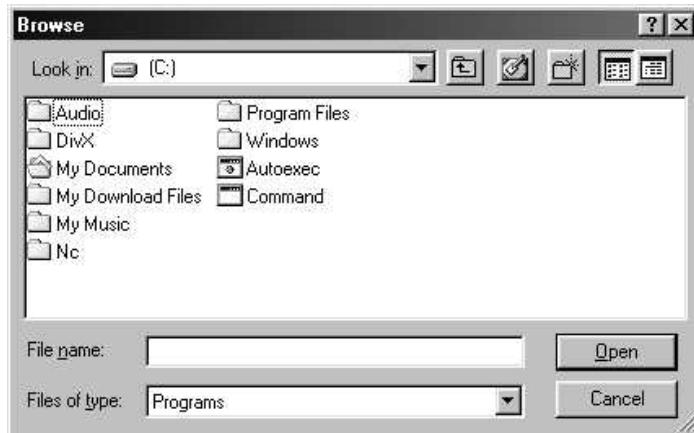


Figura 1.2. Fereastra BROWSE de selecție directă a fișierului de executat

Partea superioară a ferestrei de dialog conține caseta de selecție LOOK IN, prin care se specifică directorul curent (lista de selecție conține structura de directoare cu plasarea în sistem a directorului curent), urmată spre dreapta de butoanele UP ONE LEVEL (trecerea la directorul părinte al directorului curent), VIEW DESKTOP (în caseta de listare se afișează elementele care se găsesc pe DESKTOP), LIST (caseta de listare conține o listă cu enumerarea directoarelor și fișierelor directorului curent) și DETAILS (caseta de listare conține lista cu detaliii, adică pentru fiecare element listat se include identificarea WINDOWS - iconiță de tip și nume -, dimensiunea, dacă nu este director, tipul de fișier și data și momentul ultimei modificări).

Selecția SHUTDOWN inițiază aplicația specială de închidere sigură a sistemului calculatorului sau a platformei WINDOWS. Prin actionarea prin click a selecției SHUTDOWN se inițiază fereastra de dialog specifică, având bara de titlu formată din titlul SHUT DOWN WINDOWS și butonul de închidere a ferestrei și terminare a aplicației, fără acțiune (cu revenirea la starea anterioară inițierii), butonul X.

În partea inferioară a ferestrei de dialog se găsesc butoanele OK (realizează operația selectată din fereastră), CANCEL (închiderea ferestrei de dialog și terminarea aplicației fără acțiune) și HELP (pentru solicitarea de asistență).

Fereastra de dialog permite selecția exclusivă, prin marcare, între următoarele operații:

- STAND BY (sistemul intră în așteptare după salvarea stării curente, reducând la minim consumurile energetice; se revine la starea anterioară prin orice acționare de tastă de la tastatură sau prin orice mișcare a mouse-ului);
- SHUT DOWN (se realizează închiderea aplicațiilor uitate active, salvarea stării curente a stabilitelor din platforma WINDOWS și dezactivarea sistemului de operare; pentru calculatoarele care permit acest lucru, se poate realiza închiderea ecranului și chiar închiderea completă a sistemului de calcul - întreruperea alimentării cu energie electrică);
 - RESTART (repornirea sistemului de operare cu platforma WINDOWS);
 - RESTART IN MS-DOS MODE (repornirea sistemului de operare, dar fără utilizarea platformei WINDOWS).



Figura 1.3. Fereastra de închidere/repornire activitate, sistem de operare

I.2. Fereastra generală de aplicație

Orice activitate desfășurată într-o platformă WINDOWS se realizează printr-o fereastră de aplicație care are o construcție standardizată. Fiecare aplicație poate conține în afara ferestrei generale a aplicației și alte ferestre specifice, de lucru, și ferestre de dialog. Ferestrele de dialog sunt, la rândul lor, standardizate. De aceea, vom prezenta o fereastră de dialog la prima aplicație la care intervine, la celelalte făcând doar referire.

Fereastra generală a unei aplicații este formată dintr-o bară de titlu, o bară de meniu principal, una sau mai multe bare de SHORTCUT, pentru activitățile specifice aplicației, bara informativă și spațiul util al aplicației. Bara de titlu este prima bară a ferestrei, cea informativă este întotdeauna ultima, restul barelor aflându-se sub bara de titlu și deasupra barei informative.

Fereastra generală a aplicației poate suferi modificări relative la dimensiune și/sau poziție pe ecran. Aceste modificări influențează și ferestrele de lucru, în ceea ce privește dimensiunea și/sau nu și plasarea pe ecran a ferestrelor de dialog ale aplicației. Modificarea dimensiunii sau poziției se poate face numai pentru o fereastră la dimensiune normală (constructivă sau prestabilită).

Bara de titlu pentru fereastra generală a aplicației este formată din următoarele elemente: iconița aplicației, numele aplicației, butonul de minimizare, butonul de maximizare/restaurare și butonul de închidere a aplicației.

Iconița aplicației se află în extremitatea stângă a barei. Prin click asupra acestei iconițe se deschide o bară de selecții, prin care se pot alege acțiuni ce se efectuează la nivelul ferestrei: RESTORE (readucerea ferestrei la forma constructivă; activ doar când fereastra este maximizată), MOVE (schimbarea poziției ferestrei pe ecran; activ doar când fereastra este la dimensiune constructivă), SIZE (modificarea dimensiunii ferestrei; activ doar pentru fereastra la dimensiune constructivă), MINIMIZE (închiderea ferestrei aplicației, fără terminarea acesteia; semnalarea activării aplicației se face prin prezența butonului aplicației pe bara de START a DESKTOP-ului cu care aplicația poate fi reactivată vizibil curent cu redeschiderea ferestrei generale), MAXIMIZE (mărirea ferestrei la dimensiunea maximă a ecranului; activ doar când fereastra este la dimensiune constructivă) și CLOSE (închiderea ferestrei aplicației și terminarea execuției).

Numele aplicației sau titlul aplicației, realizează identificarea aplicației curente. În cazul în care aplicația are ferestre de lucru, dacă fereastra de lucru activă este în formă maximizată, în unele cazuri (de exemplu, WORD, EXCEL) la identificarea aplicației se adaugă și identificarea fereștei de lucru active. Numele aplicației se găsește în stânga barei de titlu, după iconița aplicației.

Butonul de închidere a ferestrei generale a aplicației și terminare a execuției (butonul X) se găsește în extremitatea dreaptă a barei de titlu. Înainte de terminarea aplicației, se acționează specific modului de lucru cu aplicația în execuție.

Butonul de minimizare (butonul -) se găsește tot în extremitatea dreaptă a barei de titlu și, fără a termina aplicația, o dezactivează provizoriu și îi închide fereastra generală (o face invizibilă). Aplicația este indicată doar prin prezența butonului aplicației pe bara de START. Acționarea acestui buton reactivează aplicația, care devine curentă, și redeschide fereastra generală (o face vizibilă), în forma anterioară minimizării.

Butonul de maximizare/restaurare are două funcții și două imagini specifice. El se află tot în extremitatea dreaptă a barei de titlu. Butonul  semnifică activitatea de maximizare și este prezent pentru fereastra la dimensiune constructivă. Prin acționarea acestei forme, fereastra este adusă la dimensiunea maximă a ecranului, butonul trecând în formă . Butonul  semnifică activitatea de restaurare și este prezent doar pentru fereastra maximizată. Prin acționarea acestui buton, fereastra este readusă la dimensiunea constructivă, butonul trecând în formă .

Forma și conținutul celorlalte elemente dintr-o fereastră generală de aplicație depind de aplicația executată și vor fi descrise la prezentarea acestora.

În afara ferestrelor generale de aplicație, platformele de tipul WINDOWS lucrează cu o serie specifică de ferestre, prin care se asigură fie dialogul curent dintre calculator și utilizator (ferestre de dialog), fie informarea utilizatorului asupra modului de desfășurare a activităților curente (ferestre informative și de decizie a utilizatorului). Aceste ferestre sunt specifice diverselor acțiuni desfășurate de om sau de mașină și vor fi prezentate în contextul în care apar.

I.3. Operații de tip editare în WINDOWS

În cadrul platformei WINDOWS se pot realiza, în diverse moduri, două feluri de operații cunoscute global sub numele de operații de editare. Ele nu acționează local pentru fiecare aplicație în parte, ci funcționează ca instrumente globale, putând fi folosite ca instrumente de comunicare între diverse aplicații activate simultan.

Instrumentul care permite aceste operații este preluat din aplicații mai vechi în care funcționa independent, și poartă numele de CLIPBOARD. El reprezintă acum o zonă de memorie comună întregii platforme WINDOWS, cu constituție dinamică (spațiul utilizat se adaptează pe parcursul desfășurării activității, în funcție de necesarul real).

Operațiile de editare se desfășoară prin intermediul unei perechi de comenzi care acționează asupra CLIPBOARD-ului.

Cea de-a doua comandă a perechii este comanda PASTE, care are ca rol transferarea informației depuse ultima dată în CLIPBORD în poziția curentă a cursorului de text din fereastra activă curent (deci în aplicația executată în mod curent). Comanda este regăsită ca atare în orice meniu de editare, dar se poate realiza și direct ca o comandă a platformei WINDOWS (combinația funcțională CRTL / V).

Numele și efectul operației de editare depind de prima comandă de acționare asupra CLIPBOARD-ului, comandă prin care se realizează un transfer de informații (selectate sau marcate) din fereastra activă la inițierea comenzi, în CLIPBOARD.

Comanda COPY realizează doar transferul specificat, fără a influența fereastra activă. Combinarea obținută prin perechea COPY-PASTE poartă numele de operație de copiere.

Comanda CUT realizează transferul specificat și elimină informația din fereastra activă. Combinarea CUT-PASTE se numește operație de mutare.

Cele două comenzi se regăsesc ca atare în orice meniu EDIT. Pentru COPY există și combinația funcțională CTRL / C realizată direct ca o comandă a platformei WINDOWS. Tot cu rol de COPY, și tot de la nivelul platformei de operare se poate realiza și un transfer special către CLIPBOARD, operație în care este implicată chiar imaginea grafică a întregii ferestre active curent pe DESKTOP. Acțiunea este obținută prin combinația funcțională ALT / PRINT SCREEN, PRINT SCREEN, fiind prima tastă aflată la dreapta tastelor funcționale F1-F12 (respectiv F1-F14, dacă este cazul).

I.4. WINDOWS EXPLORER

Aplicația WINDOWS EXPLORER este instrumentul principal de navigare în structura logică a unui calculator modern. De asemenea, prin intermediul acestui program se realizează principalele operații asupra echipamentelor conectate la calculator.

Există două modalități de prezentare:

- formatul normal; spațiul util este împărțit în două subferestre, una stângă de prezentare a structurii logice a calculatorului și una dreaptă de listare a unui punct de structură, considerat deschis;
- formatul pagină WEB; spațiul util este, de asemenea, împărțit în două subferestre, numai că subfereastra stângă prezintă, de această dată, doar punctul de structură deschis explorării împreună cu o parte a proprietăților sale sau ale selecțiilor realizate asupra conținutului său.

Deoarece modalitățile de lucru în cele două prezentări sunt similare, însă formatul normal oferă mai multe facilități, vom face prezentarea asupra acestui format (figura 1.4).

Subfereastra de structură este o afișare a structurii logice, parțială sau integrală, structurată arborescent. Fiecare punct al arborescenței reprezintă o noțiune logică legată de componentele logice ale calculatorului, echipamentele fizice conectate (nu neapărat și active) și directoarele și subdirectoarelor echipamentelor fizice. Componenta supusă explorării (deschisă) este semnalată prin schimbarea culorii folosite pentru fond și caractere.

Conectarea componentelor în arborescență se face în trei moduri:

- 1) legare simplă, semnificând un punct al structurii logice pentru care nu există subdirectoare;
- 2) cuplare prin pătrat cu semn +; în acest caz, elementul conține subdirectoare, dar acestea nu sunt vizibile; pentru a realiza și afișarea acestora se acționează cu click asupra pătratului;
- 3) conectare prin pătrat cu semn -; componenta conține subdirectoare și acestea sunt afișate în continuare; dacă nu se mai dorește prezența lor în structura vizibilă, se acționează cu click asupra pătratului.

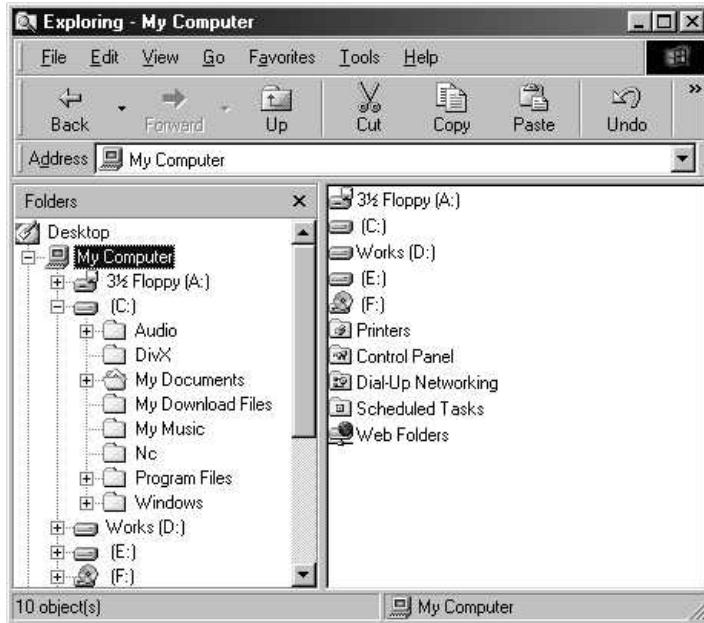


Figura 1.4. Fereastra generală pentru aplicația Explorer

Subfereastra din dreapta prezintă o listă a tuturor componentelor punctului de structură deschis: echipamente, directoare și fișiere. Acționarea prin dublu click asupra unui element afișat are ca efect:

- accesarea echipamentului sau directorului care astfel devine deschis;
- execuția fișierului, dacă elementul este un fișier program sau comenzi indirecțe;
- execuția aplicației prin care fișierul este recunoscut și încărcarea acestuia în aplicație.

Selectarea unei componente se face prin simpla poziționare a cursorului mouse pe element. Selectarea multiplă se realizează prin apăsarea continuă a butonului stâng al mouse-ului și deplasarea acestuia pentru a cuprinde în chenarul activat componentele dorite. Selectarea tuturor elementelor afișate se poate face prin acționarea activității SELECT ALL din meniul EDIT. Deselectarea se produce prin acționarea cu click în orice punct din zona neselectată.

Activități legate de echipamente

Selectarea unui echipament și acționarea cu click cu butonul drept al mouse-ului produc deschiderea unei bare verticale de meniu care conține operațiile care se pot realiza asupra echipamentului.

Un astfel de proces este cel de pregătire a echipamentului pentru lucru, aplicabil pentru dischete și harddisk-uri (formatare). Această operație (selecția FORMAT) se poate desfășura doar dacă echipamentul nu este accesat de nici o aplicație activată curent. Operația nu este indicată decât pentru dischete și se realizează prin intermediul unei ferestre specifice (figura 1.5.). Diferența față de formatarea sub sistemul de operare

MS-DOS constă în posibilitatea de a alege între formarea integrală și cea rapidă (doar eliminarea conținutului suportului prin recrearea fișierelor specifice).

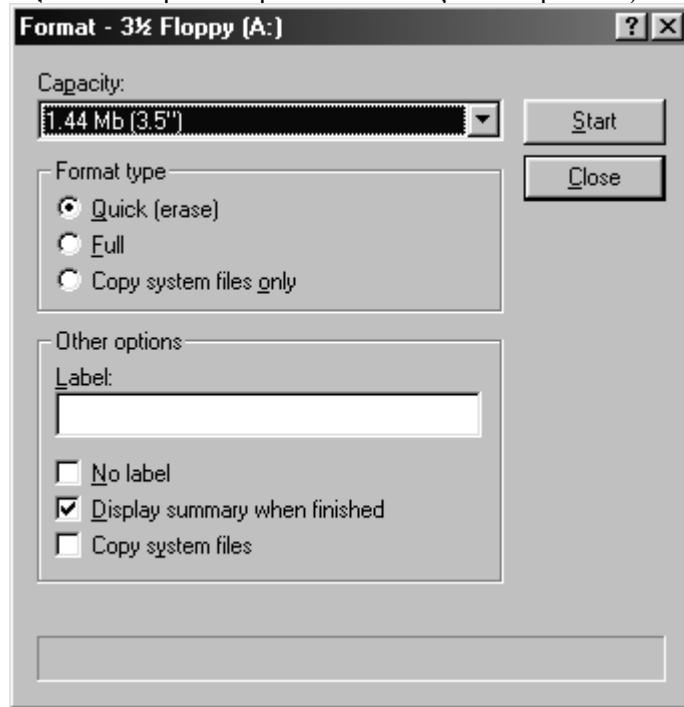


Figura 1.5. Fereastra FORMAT

Un alt proces posibil aplicabil doar pentru echipamentele de floppy disk este duplicarea unei dischete (selecția COPY DISK). Se realizează prin intermediul ferestrei din figura 1.6. și funcționează la fel cu modul de lucru al comenzi DISKCOPY din sistemul de operare MS-DOS.

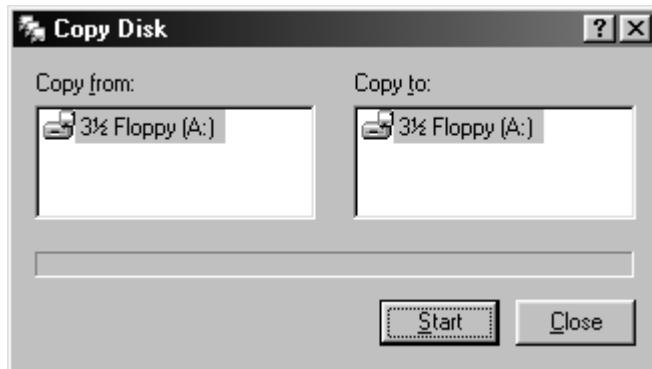


Figura 1.6. Fereastra duplicare dischetă

Activități legate de directoare și fișiere

Operațiile legate de directoare și fișiere se desfășoară global, fără a se face distincție între cele două tipuri de componente. Acțiunile cuprind activități realizate fără selecție de elemente (crearea directă a directoarelor sau a fișierelor specifice aplicațiilor existente pe calculator) și operații asupra componentelor selectate.

În operația de creare a fișierelor nu se face accesul la aplicația implicată, ceea ce face ca timpul necesar realizării comenzi să fie foarte scurt. După creare, componenta are un nume implicat, NEW FOLDER pentru directoare și numele documentului vid din aplicația solicitată. Zona de identificare este lăsată deschisă, cursorul de test fiind prezent și permitând utilizatorului stabilirea numelui efectiv.

Activitățile aplicate elementelor selectate privesc:

- copierea și mutarea;
- ștergerea;
- schimbarea identificării;
- afișarea proprietăților.

Schimbarea identificării nu se poate aplica decât asupra selecției unui singur element și are ca efect deschiderea zonei de nume, prezența cursorului text indicând posibilitatea introducerii unei identificări noi. Pentru ca numele nou să devină efectiv, trebuie ca fișierul implicat să nu fie accesat în nici o aplicație activă.

Copierea și mutarea sunt operații generale de tip editare și au fost prezentate în cadrul acestui capitol.

Ștergerea realizată la nivelul harddisk-urilor nu este una definitivă, ci are ca efect doar transferarea componentelor selectate într-un director special, RECL YCLED, cu rol de coș de gunoi, existând câte un astfel de director pe fiecare echipament de harddisk. Dacă ștergerea se aplică dischetelor, ea este definitivă.

Pentru operația de copiere există și o comandă specială, SEND TO, cu specificarea destinației aleasă din meniu specific.

Operațiile de copiere și mutare se pot executa și direct, prin deplasarea cursorului mouse, fie în aceeași fereastră de explorare, fie între două ferestre de explorare deschise simultan. La lucru cu o singură fereastră operațiile se pot realiza fie în cadrul ferestrei de listare (transferul din directorul curent într-un subdirector al său), fie între fereastra de listare și cea de structură (transferul din directorul curent în directorul/echipamentul care devine selectat prin poziționarea cursorului mouse). Deplasările se realizează cu apăsarea continuă a unuia din butoanele mouse-ului; acțiunea efectivă depinde de butonul utilizat, astfel:

- Utilizarea butonului stâng. La eliberarea butonului se realizează automat operația de mutare, dacă transferul se realizează pe aceeași unitate de echipament sau de copiere, dacă sunt implicate două echipamente cu identificări diferite.
- Utilizarea butonului drept. La eliberarea butonului se generează o bară de meniu vertical, din care utilizatorul poate selecta acțiunea dorită: MOVE HERE (pentru mutare), COPY HERE (pentru copiere) sau CANCEL (renunțarea la realizarea operației).

Operațiile realizate pot fi anulate imediat după realizarea lor, prin comanda UNDO, prezentă în meniul EDIT.

II. EDITARE DE TEXT

II.1. Prezentare generală

Principalele facilități oferite, care se regăsesc și în componente meniurilor, sunt:

- în stabilirea/modificarea caracteristicilor scierii;
- formatarea facilă a paginării în funcție de cerințele de paginare;
- posibilitatea de paginare tipografică, prin scrierea pe mai multe coloane;
- realizarea antetelor și subsolurilor de pagină, repetate automat în funcție de paginarea aleasă;
- introducerea în text, prin inserție, a unor caractere speciale și obiecte, cum ar fi imagini, desene, grafice, ecuații, realizate atât independent, cât și prin intermediul unor programe externe lansate în execuție în interiorul editării curente;
- prelucrarea în tabele, inclusiv realizarea unor calcule elementare în acestea;
- asigurarea unor instrumente speciale de ușurare a activității: verificarea lexicală și gramaticală, autorezumarea, autocorectarea, realizarea documentelor multiple cu formă fixă pe baza unei surse de date.

II.2. Caracteristici de scriere

Caracteristicile de scriere sunt elemente legate de formatul scierii efective, a modului de formare a rândurilor, paragrafelor și paginilor. O parte a acestor caracteristici pot fi stabilite direct, prin intermediul barei de meniu FORMATTING, care are formatul din figura 2.1.



Figura 2.1. Bara de meniu FORMATTING

Caracteristicile pentru scrierea efectivă pot fi stabilite prin intermediul comenzi FONT din meniul FORMAT. Prin fereastra prezentată în figura 2.2. se pot stabili nu numai caracteristicile pentru caracterele utilizate în scriere (pagina FONT), ci și spațierea dintre caractere (pagina CHARACTER SPACING) și posibilitatea realizării animației pentru textul scris (pagina ANIMATION). Pentru selecțiile realizate, în fiecare pagină, se prezintă exemplificarea rezultatului selecției.

Structura paragrafelor normale se stabilește prin intermediul comenzi PARAGRAPH din meniul format, prin care se deschide fereastra cu același nume (figura 2.3). Construcția uzuală este cea relativă la spațiere, prin care se indică pentru un paragraf selectat atât distanțele față de cele vecine, cât și distanța dintre rândurile care formează paragraful.

O serie de paragrafe pot fi înlanțuite în liste numerotate sau cu fiecare element marcat. Aceste construcții speciale se obțin prin comanda BULLETS AND NUMBERING. Urmarea acțiunării comenzii este deschiderea ferestrei cu același nume (figura 2.4.).

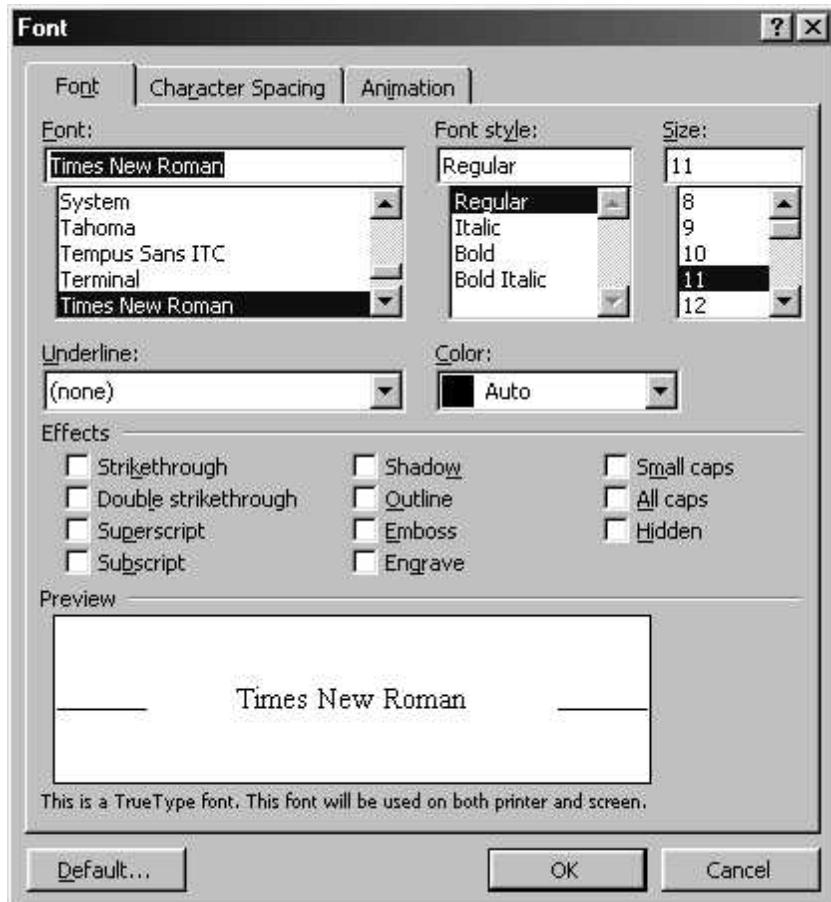


Figura 2.2. Fereastra FONT

Construcțiile de forma marcată se realizează având caracterele de marcat la același nivel, caractere alese prin pagina BELLETED. Listele numerotate, cu valorile de numerotare de același tip și aflate la același nivel se specifică prin intermediul paginii NUMBERED.

Formele complexe permit realizarea listelor de liste, cu schimbarea modalității de numerotare și/sau marcare. Specificațiile pentru aceste configurații de paragrafe se indică în pagina OUTLINE NUMBERED.

Fiecare alegere făcută, din oricare din cele trei pagini, poate fi modificată, conform dorinței utilizatorului, prin utilizarea butonului CUSTOMIZE.

Modul de formare al paginii poate conține zone repetabile automat la fiecare pagină, sau la fiecare pagină cu aceeași paritate. Accesul în aceste zone se poate face prin intermediul comenzii HEADER AND FOOTER din meniu VIEW. Scrierea în aceste zone se realizează în același mod ca scrierea din document. Revenirea la textul normal se face printr-un click în spațiul document sau prin utilizarea comenzii CLOSE din bara de meniu deschisă o dată cu vizualizarea zonelor repetabile (figura 2.5.).

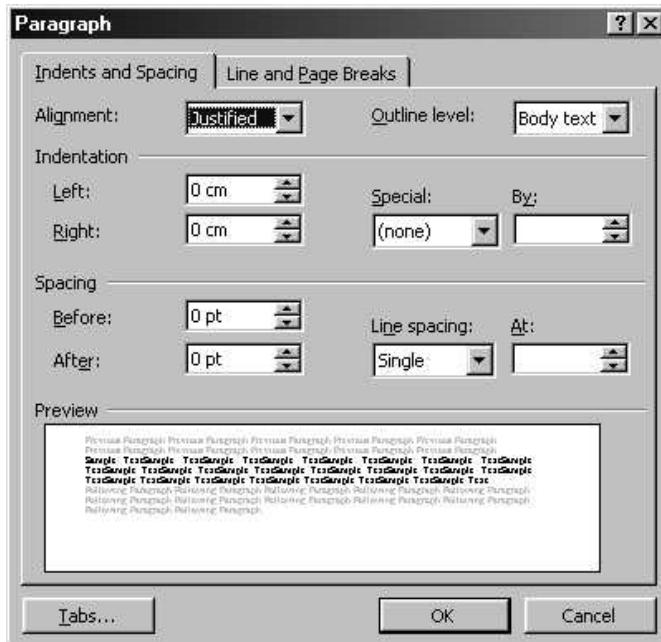


Figura 2.3. Fereastra PARAGRAPH

Un element foarte util pentru realizarea documentelor de dimensiuni mari este numerotarea paginilor. Acest lucru se realizează în cadrul zonelor fixe și poate fi introdus din bara de meniu HEADER AND FOOTER sau poate fi controlată direct prin comanda PAGE NUMBERS din meniu INSERT. Prima specificare a paginării se realizează prin fereastra PAGE NUMBERS (figura 2.6.), prin care se indică poziția și modul de aliniere a numărului de pagină și dacă acesta apare sau nu pe prima pagină a documentului. Trecerea la cea de-a doua etapă a specificațiilor relative la numerotarea paginilor se realizează prin acționarea butonului FORMAT.

Prin acționarea butonului, se deschide fereastra PAGE NUMBER FORMAT pentru specificarea modalității de numerotare și a valorii de început pentru numerotare (figura 2.7.).

II.3. Caracteristici de paginare

Paginarea cuprinde două aspecte, unul legat de formatul logic pentru fiecare pagină relativ la construcția ei, și un aspect fizic privind posibilitatea de imprimare a paginilor pe hârtie.

Caracteristicile logice se stabilesc prin utilizarea comenzii BORDERS AND SHADING, prin care se deschide fereastra cu același nume (figura 2.8.)

conținând pagini separate pentru indicarea linierilor care se aplică la spațiul util (BORDERS), a celor pentru pagina fizică (PAGE BORDER) și a culorii de fond pentru întreaga pagină (SHADING).

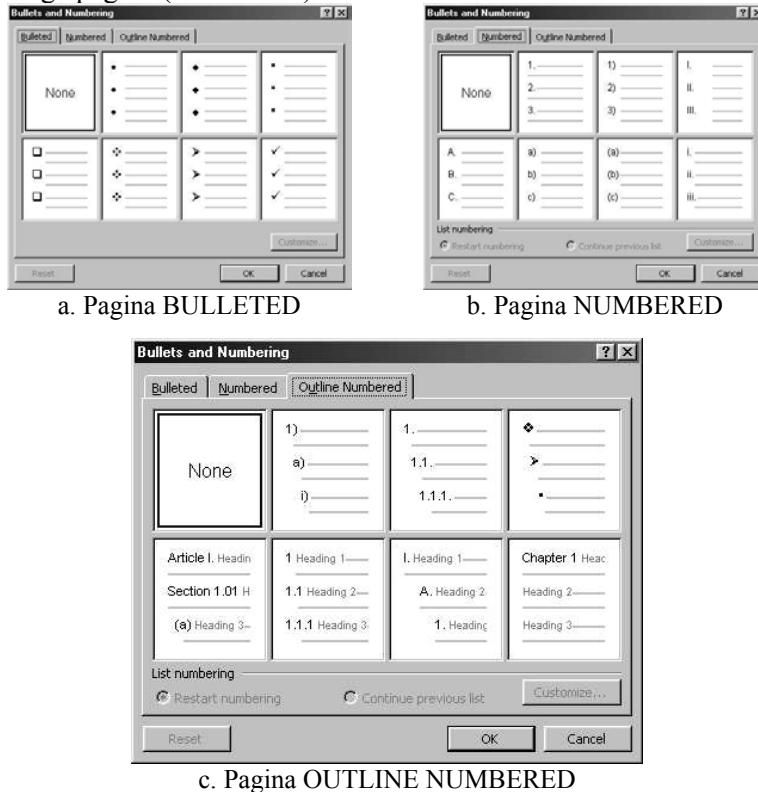


Figura 2.4. Fereastra BULLETS AND NUMBERING



Figura 2.5. Bara de meniu HEADER AND FOOTER



Figura 2.6. Fereastra PAGE NUMBERING

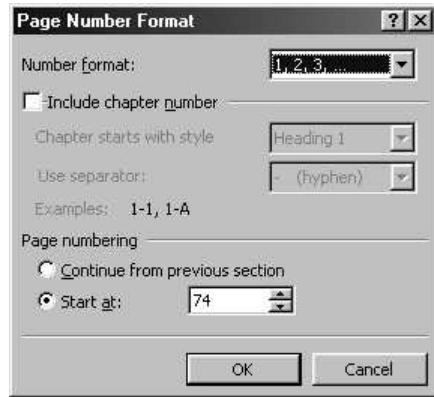


Figura 2.7. Fereastra PAGE NUMBER FORMAT

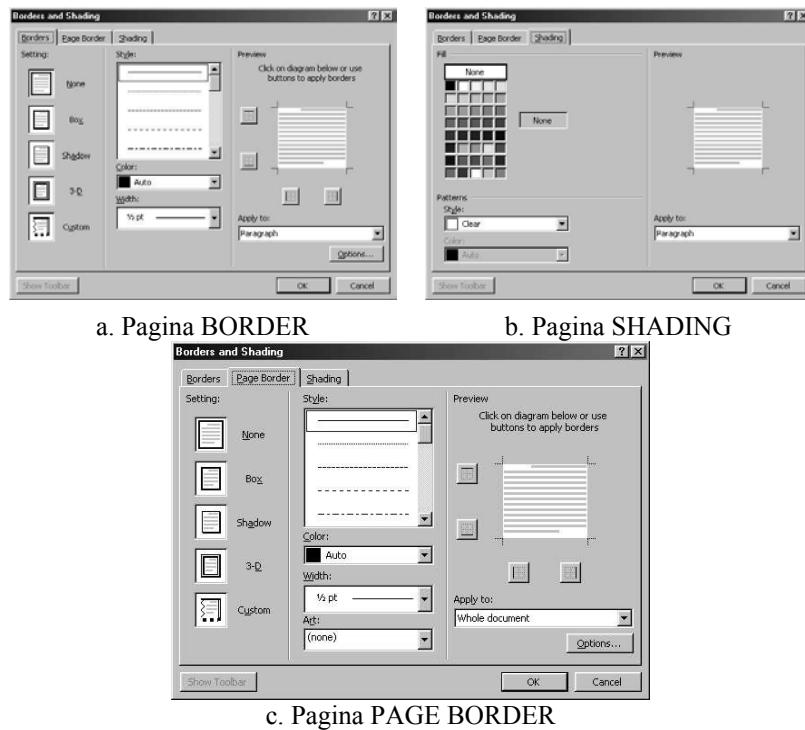
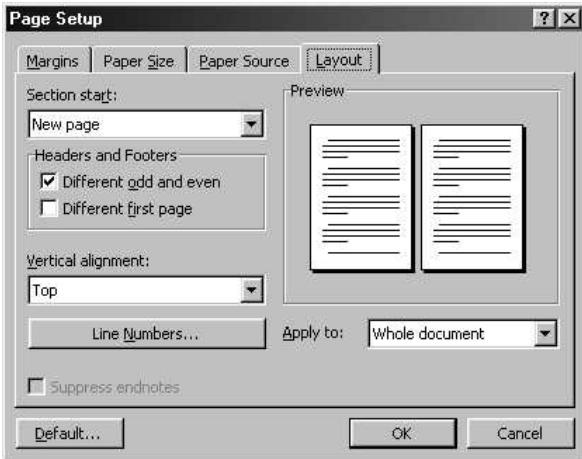
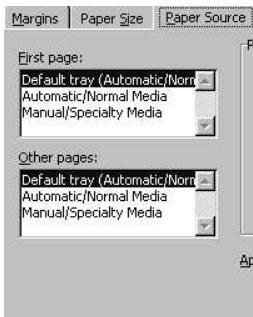


Figura 2.8. Fereastra BORDERS AND SHADING

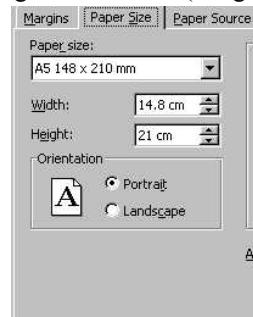
Aspectul fizic al paginării este asigurat prin utilizarea comenzii PAGE SETUP din meniul FILE. Prin solicitarea acestei activități se activează fereastra cu același nume (figura 2.9.).



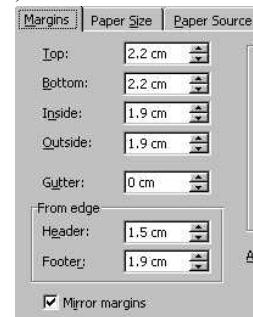
a. Pagina LAYOUT (integrală)



b. PAPER SOURCE



c. PAPER SIZE



d. MARGINS

Figura 2.9. Fereastra PAGE SETUP

Fereastra este constituită din patru pagini distincte, care au comună partea dreaptă formată din vizualizarea rezultatului selecțiilor curente (PREVIEW) și de o casetă de indicare a ţinței pentru acestea (APPLY TO).

Pagina MARGINS este orientată pentru a specifica, în unitatea de măsură în care se lucrează curent, spațiile care vor fi lăsate între text și marginea fizică a unei pagini de hârtie. În plus, se definesc dimensiunile pentru zonele fixe (antet și subsol), minimale pentru pagina logică. De asemenea, în partea finală a paginii se indică prin selecție modul de formare a paginilor (normal sau pentru imprimare față-verso, „în oglindă” sau MIROR).

Pagina PAPER SIZE se utilizează pentru a specifica dimensiunile fizice ale hârtiei, în exprimarea tipografică și/sau în dimensiuni reale și orientarea utilizată pentru pagină (PORTRAIT sau LANDSCAPE). Pagina PAPER SOURCE are rol de a indica modul de încărcare a paginilor la momentul realizării operațiilor de imprimare.

Pagina LAYOUT permite indicarea modalității în care se încep secțiunile, modalitatea în care se aplică formarea antetelor și subsolurilor care apar efectiv, alinierea verticală în cadrul paginilor și eventuale informații de numărare a rândurilor din fiecare pagină.

II.4. Introducerea în text a caracterelor speciale

Redactarea normală folosește drept caractere cele care au corespondent direct pe tastatura calculatorului. Există însă numeroase cazuri în care este necesară introducerea în text a unor semne speciale, care nu se regăsesc pe tastatură.

Această operație se poate realiza în două moduri, în funcție de frecvența lucrului cu un caracter particular. Dacă un caracter se utilizează rar, atunci pentru introducerea sa în text se poate folosi inserția directă prin intermediul comenzi SYMBOL din meniu INSERT. Această comandă dă controlul unei ferestre cu același nume (figura 2.10.) pentru alegerea caracterului dorit și introducerea lui în text în poziția curentă a cursorului text.

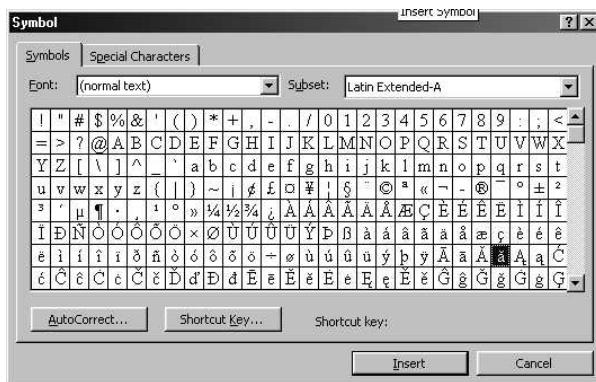


Figura 2.10. Fereastra SYMBOL

Dacă un caracter care nu se găsește pe tastatură este utilizat frecvent, modalitatea de lucru specificată mai sus este incomodă. Pentru acest caz, programul WORD oferă utilizatorului posibilitatea de atașare la caracter a unei combinații funcționale de taste, urmând ca ulterior pentru introducerea caracterului respectiv să se folosească această combinație și nu inserarea directă.

Stabilirea combinației funcționale de taste se face din fereastra SYMBOL, prin utilizarea butonului SHORTCUT KEY. Prin acționa-rea lui se deschide fereastra CUSTOMIZE KEYBOARD (figura 2.11.), prin care:

- se specifică o combinație funcțională de taste pentru caracterul selectat curent din fereastra SYMBOL;
- se reține această combinație fie în fișierul general de setări (NORMAL), fie într-un fișier de setări legat direct de documentul editat (un fișier cu același nume, dar cu extensie diferită);
 - se exclude o combinație funcțională atașată anterior caracterului;
 - se elimină toate combinațiile funcționale atașate curent pentru caracterul selectat din fereastra SYMBOL.

II.5. Inserarea obiectelor

În interiorul unui document poate fi necesară introducerea unor elemente de factură specială, care nu pot fi realizate prin intermediul facilităților oferite de programul WORD. Pentru astfel de elemente, este necesară folosirea unor programe speciale.

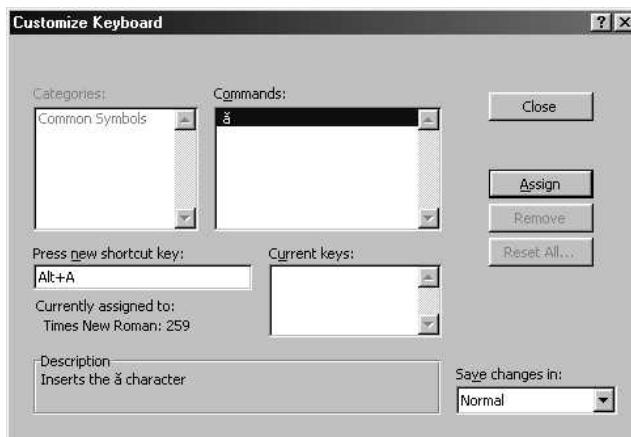


Figura 2.11. Fereastra CUSTOMIZE KEYBOARD

O soluție posibilă ar fi activarea separată a acestor programe, realizarea construcțiilor dorite cu salvarea specifică și apoi revenirea la editorul de texte pentru a prelua aceste construcții. Modul de lucru indicat impune cunoașterea acceptabilă a aplicației implicate de necesitățile utilizatorului, un consum suplimentar de spațiu, consum mai mare de timp datorat operațiilor de acces la periferice pentru salvare/încărcare. O problemă importantă care poate interveni este compatibilitatea dintre ieșirile aplicației folosite și intrările posibile pentru programul WORD.

Soluția pusă la dispoziție de editorul de texte WORD este de a considera construcțiile speciale drept obiecte (externe) și a le realiza din interiorul programului, prin intermediul unor ferestre de lucru specifice sau casete speciale în document, prin apelarea directă a aplicației dorite. Alegerea instrumentului de lucru se face la inițierea procesului se inserare a unui obiect, după acționarea comenții OBJECT din meniul INSERT, prin intermediul ferestrei OBJECT (figura 2.12.).

II.6. Utilizarea tabelelor

O facilitate importantă pusă la dispoziție de programul WORD este legată de editarea sub formă de tabele, inclusiv posibilitatea de a realiza și unele calcule matematice elementare.

Un tabel este structurat în celule grupate pe linii și pe coloane. Celulele pot fi identificate prin combinarea unei identificări de coloană și de linie. Pentru identificarea coloanelor se folosește o numerotare alfabetică, prima coloană fiind identificată prin A, a două prin B etc. Pentru identificarea liniilor, se folosește numerotarea cu valori

naturale (prima linie are identificarea 1, a două 2 etc.). Astfel, o celulă aflată în coloana 3 și linia 5 a unui tabel poate fi identificată prin „C5”.

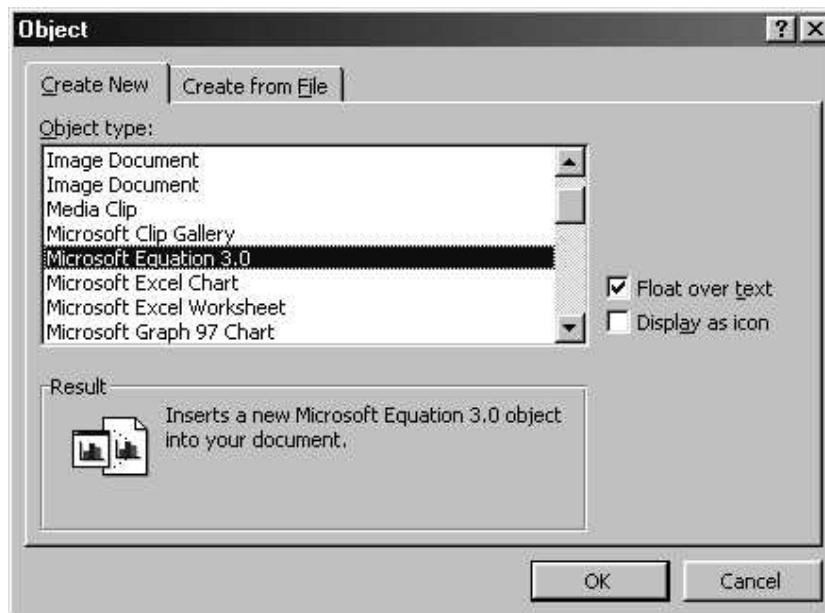


Figura 2.12. Fereastra OBJECT

Fiecare celulă este văzută ca o entitate de sine stătătoare și are propriile caracteristici de scriere, funcționând asemănător cu partea utilă, de scriere efectivă, a unei pagini. Editarea tabelelor are dedicată o componentă specială de meniu (TABLE).

Construcția unui tabel poate fi asigurată manual, prin desenare (comanda DRAW TABLE), sau automat (INSERT TABLE). La construcția automată, specificarea caracteristicilor tabelului introdus în document în poziția curentă a cursorului se face prin fereastra INSERT TABLE (figura 2.13.). Se aleg numărul inițial de linii și de coloane și dimensiunea orizontală a coloanelor. Celulele obținute în tabel sunt toate de aceeași dimensiune.

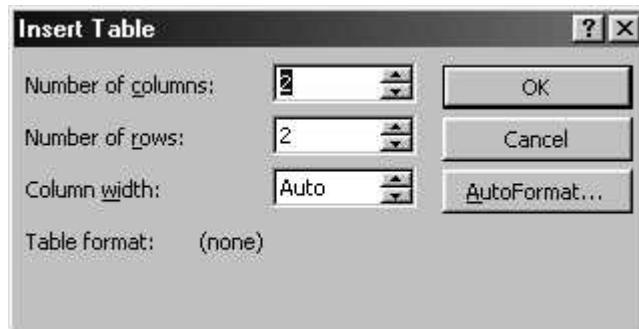


Figura 2.13. Fereastra INSERT TABLE

Configurarea tabelelor

Operațiile care pot fi grupate sub această titulatură cuprind reunirea și divizarea celulelor, ștergerea și inserarea de celule, linii sau coloane, stabilirea dimensiunilor pentru linii și coloane.

Pentru fiecare din aceste operații, există în cadrul meniului câte o comandă specifică. Astfel:

- MERGE CELLS produce transformarea unui grup de celule selectate într-o singură;
- SPLIT CELLS subîmparte o celulă, operația efectivă fiind precedată de intervenția ferestrei cu același nume, pentru specificarea numărului de coloane și linii rezultante prin divizare;
- ștergerea se realizează prin comenzi formate din cuvântul DELETE, urmat de cuvântul de specificare a elementelor supuse ștergerii, CELLS (pentru celule), ROWS (linii) și COLOMNS (coloane); în mod similar, se formează și comenziile pentru inserare, având ca prim constituent cuvântul INSERT; de asemenea, la nivelul liniilor, coloanelor și întregului tabel (TABLE) se formează comenziile pentru selectare (SELECT).

Stabilirea dimensiunilor pentru linii și coloane se realizează prin intermediul comenzi CELL HEIGHT AND WIDTH care activează fereastra cu același nume (figura 2.14.).

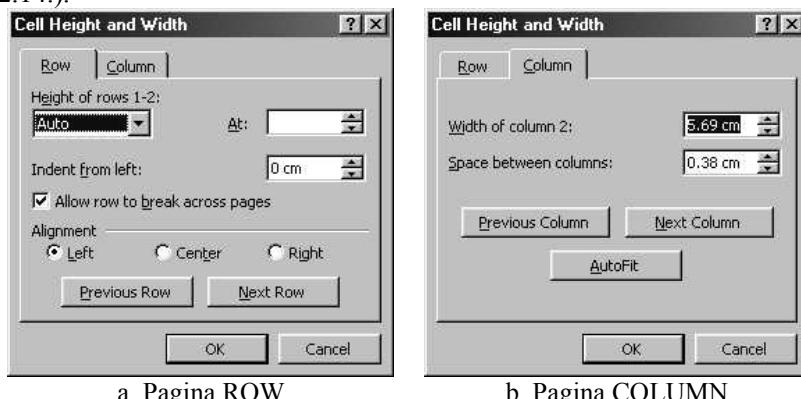


Figura 2.14. Fereastra CELL HEIGHT AND WIDTH

Ambele pagini ale ferestrei permit trecerea de la coloana/linia curentă la cea anterioară/următoare, lucru care permite stabilirea dimensiunilor din aproape în aproape, fără ieșirea din fereastră.

Ordonarea informațiilor din tabele

Ordinea de prezentare a datelor introduse în tabele nu este restricționată de cea în care acestea au fost scrise. De asemenea, utilizatorul poate modifica ordinea de prezentare, fără să fie obligat la realizarea unor operații complicate de transfer. Aceste lucruri sunt posibile datorită instrumentului de „sortare” a informațiilor, care este aplicabil la nivelul tabelelor (integral sau pe zonă selectată).

Sistemul de execuție a ordonării permite să se specifice dacă tabelul sau partea de tabel supusă operației conține ca primă linie un „cap de tabel”, caz în care această linie nu este considerată la acționarea asupra tabelului, în schimb este utilă pentru identificarea mai exactă a coloanelor reprezentând criteriile de lucru. Dacă nu se specifică prezența unui cap de tabel, identificarea coloanelor se face prin COLUMN, urmat de numărul coloanei. Dacă s-a ales varianta cu prezența capului de tabel, identificarea coloanelor se face prin informația regăsită în prima celulă a coloanei.

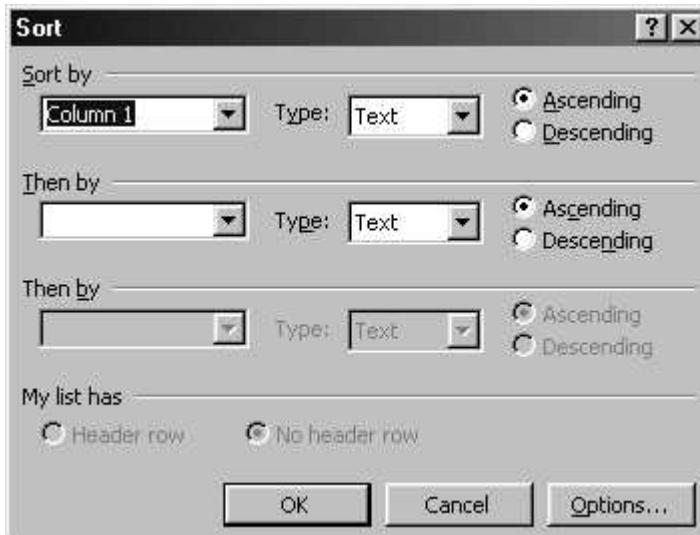


Figura 2.15. Fereastra SORT

Un criteriu de ordonare este dat de identificarea coloanei, de tipul informației considerate a forma celulele coloanei și de ordinea considerată – crescătoare (ascendentă) sau descrescătoare (descendentă). Pentru ordonare se pot utiliza până la trei criterii de sortare. Cel de-al doilea criteriu, dacă este prezent, se aplică doar pentru a face o ordonare a liniilor pentru care valorile corespunzătoare primului criteriu sunt egale. În mod similar, dacă sunt date trei criterii, ultimul se aplică doar pentru liniile în care valorile pentru primele două criterii sunt egale.

Trecerea la operația de ordonare se face prin comanda SORT, care inițiază dialogul de stabilire a criteriilor prin fereastra cu același nume (figura 2.15.), sortarea efectivă intervenind după acționarea butonului OK din fereastră.

Realizarea calculelor în tabele WORD

În interiorul tabelelor pot fi realizate calcule elementare, cu ajutorul unor formule matematice în care operanții pot fi: valori din celule, constante sau funcții definite intern programului WORD.

Unele din funcții pot avea drept argumente și mulțimi de celule grupate în zone dreptunghiulare. Pentru specificarea acestora, se utilizează, în general, o construcție formată din identificările celulelor din colțurile stânga sus – dreapta jos, separate de caracterul ":". Pentru zone speciale, se pot folosi și unele cuvinte cheie de definire a zonelor, LEFT, pentru toate celulele aflate la stânga celei în care se specifică formula de

calcul și ABOVE, pentru toate valorile aflate în celulele de deasupra celei care conține formula. Automat se consideră că se dorește realizarea sumei valorilor ABOVE. Dacă nu există nici o valoare ABOVE, se consideră sumarea valorilor LEFT. Dacă nu există nici valori ABOVE și nici LEFT, formula inițială este considerată vidă (formată doar din caracterul ei de început).

La realizarea calculelor în tabele Word, modificarea unei valori implicate în formule nu produce și refacerea calculelor, acest lucru necesitând reintrarea în sistemul de specificare a formulelor, care este activat prin comanda FORMULA din meniul TABLE. Specificarea formulelor se face prin fereastra cu același nume (figura 2.16.).

Fereastra permite, pe lângă introducerea efectivă a formulei (care trebuie să înceapă prin caracterul „=”), specificarea formatului de scriere a rezultatului și o casetă prin intermediu căreia se poate selecta funcția dorită, din cele puse la dispoziție de programul WORD.

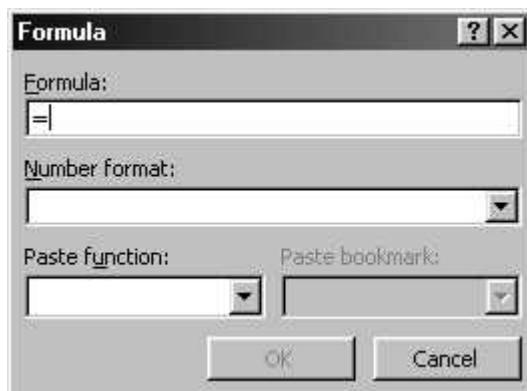


Figura 2.16. Fereastra FORMULA

II.7. Documente interconectate

Unul din cele mai utile instrumente de lucru pus la dispoziție de programele de tip editoare de texte este interconectarea documentelor (MAIL MERGE), realizat inițial pentru ușurarea activităților de transmitere a corespondenței cu formă fixă (de unde și numele său).

Sistemul de lucru presupune existența a trei elemente pentru obținerea rezultatului final: un document primar, o sursă de date și un sistem de selectare a datelor folosite. Toate aceste elemente pot fi construite doar prin sistemul de interconectare a documentelor, activat prin comanda MAIL MERGE din meniul TOOLS. Activarea acestei operații are ca efect și adăugarea la sistemul de bare de meniu a barei cu shortcut-urile specifice (figura 2.17).

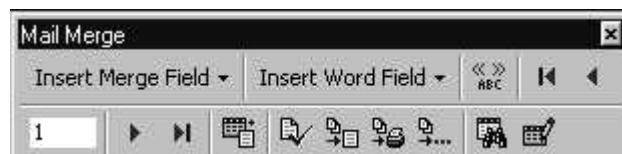


Figura 2.17. Bara de meniuri MAIL MERGE

Documentul primar reprezintă o formă fixă pentru un singur document final, realizat pentru un grup de date individuale. În acest document intervine o legătură către sursa de date, realizată formal prin „variabile”.

Sursa de date poate fi preluată din sistemele de tipul baze de date accesibile sau poate fi creată în sistemul MAIL MERGE, caz în care va fi salvată separat într-un document WORD sub formă de tabel, cu sau fără cap de tabel. Dacă sursa de date nu are cap de tabel, acesta trebuie reținut, la rândul său, ca document WORD, într-un fișier separat.

Capul de tabel al sursei de date definește formal (ca variabile) tipurile informației conținute pe fiecare coloană. Un grup individual de date conține toate informațiile aflate pe o linie a tabelului sursă de date și poartă numele de înregistrare (articol). Fiecare celulă dintr-o linie a tabelului reprezintă o dată particulară din cadrul înregistrării și poartă numele de „câmp” al înregistrării.

Datele dintr-o înregistrare pot fi folosite, integral sau parțial, pentru realizarea documentelor individuale finale și nu există restricții asupra numărului de câmpuri și a conținutului acestora. Există o situație de excepție, în care pentru realizarea operației finale unul din câmpurile din articol trebuie să conțină o valoare de factură specială.

Ultimul element care trebuie considerat este un sistem special, numit sistem de cereri (queries). Acest sistem este optional, absența specificării lui producând utilizarea tuturor articolelor din sursa de date pentru obținerea documentelor finale, înregistrări considerate exact în ordinea în care au fost introduse.

Prezența sistemului de cereri permite:

- specificarea unor criterii de alegere a articolelor care vor fi utilizate în faza de obținere a documentelor finale;
- utilizarea articolelor într-o anumită ordine, indicată prin criterii de ordonare similară celor folosite pentru sortarea datelor din-tr-un tabel WORD.

Pentru documentele primare, există mai multe variante de format, fiecare cu o utilizare specifică. Formatul este ales încă de la intrarea în sistemul MAIL MERGE pentru crearea documentului primar, operație realizată cu o fereastră de asistență (MAIL MERGE HELPER – figura 2.18.), ce poate fi activată din bara de meniuri MAIL MERGE de câte ori se consideră necesar. Fereastra conține enumerate toate cele trei elemente descrise mai sus.

Imaginea dată pentru fereastra de asistență este una maximală. La intrarea în sistemul de interconectare a documentelor în fereastră apar numai butoanele din partea stângă, iar singurul buton activat este cel din segmentul documentului primar (CREATE). La încărcarea unui document WORD dintr-un fișier, dacă acesta corespunde unui document primar MAIL MERGE, sistemul de interconectare este activat automat.

Formatele care se pot alege pentru documentul primar sunt: scrisoa-re (FORM LETTERS), etichetă poștală (MAILING LABELS), plic poștal (ENVELOPES) și catalog. La alegerea formatului dorit, după acționa-rea componentei, se realizează crearea documentului primar, posibilă atât în documentul activ curent, cât și într-un document nou. Existența documentului primar produce, în fereastra de asistență, apariția butonului de EDIT al zonei 1 (pentru trecerea la editarea documentului primar) și activarea butonului GET DATA al zonei pentru sursa de date.

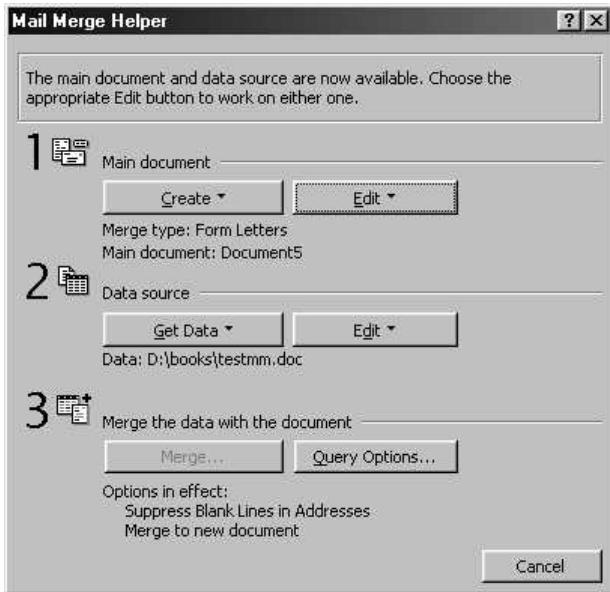


Figura 2.18. Fereastra MAIL MERGE HELPER

Pentru indicarea sursei de date se folosește butonul GET DATA de care sunt atașate variantele de: creare a unei surse de date noi (CREATE), deschiderea unei surse de date existente (OPEN), utilizarea ca sursă de date ale altor fișiere structurate pentru date (NOTE BOOKS, baze de date) și specificarea capului de tabel separat de sursa de date. Existenta sursei de date, chiar fără a conține date efective, produce în fereastra de asistență apariția butonului EDIT al zonei 2 și a butonului QUERIES din zona 3.

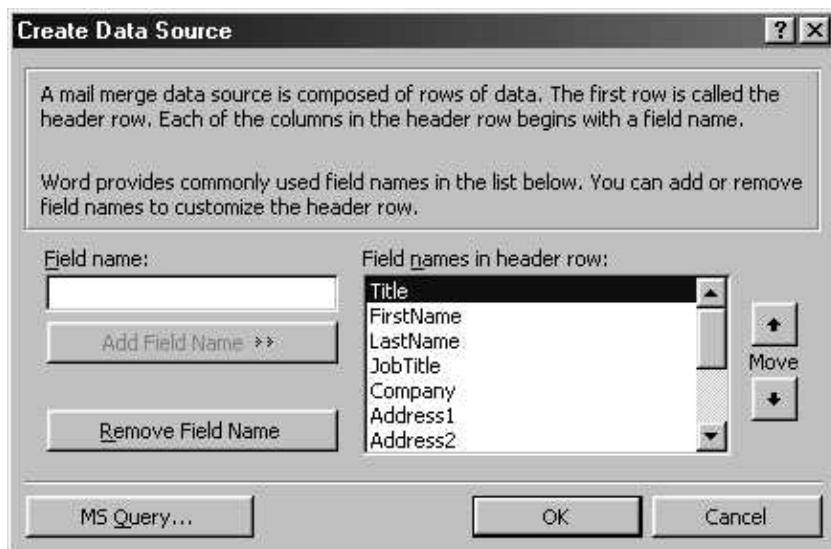


Figura 2.19. Fereastra CREATE DATA SOURCE

Sursa de date pentru sistemul de interconectare a documentelor este organizată ca structură prin intermediul ferestrei CREATE DATA SOURCE (figura 2.19.). Structura implicită conține informațiile relative la corespondență în versiunea US. Eliminarea unui câmp din cele existente se realizează prin poziționarea pe acesta și acționarea butonului REMOVE FIELD NAME, iar adăugarea unui câmp nou prin ADD FIELD NAME. Adăugarea unui câmp se realizează printr-o identificare simbolică a informației ce va fi conținută în câmp. Din această cauză, introducerea din caseta FIELD NAME trebuie să fie un identificator în sensul celor folosite în limbajele de programare și baze de date (un sir compact de caractere litere, cifre și caracterul „_”).

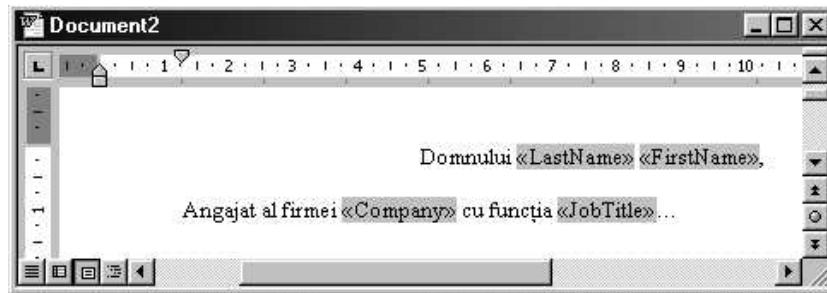


Figura 2.20. Exemplu de document primar

Editarea documentului primar reprezintă crearea formei pentru documentele care se vor obține în final și în care părțile fixe sunt considerate cele redactate normal. În documentul primar, prin utilizarea shortcut-ului INSERT MERGE FIELD, se introduc legăturile la sursa de date prin intermediul identificărilor selectate din lista celor disponibile, afișate pe o bară verticală după acționarea shortcut-ului. Nu este obligatorie utilizarea tuturor câmpurilor existente în definirea structurii sursei de date. Identificatoarele informațiilor care vor fi preluate din sursa de date sunt semnalate prin prezența lor între paranteze unghiulare duble (figura 2.20.). Existența a cel puțin unei legături la sursa de date în documentul primar produce și activarea butonului MERGE din zona 3 a ferestrei de asistență.

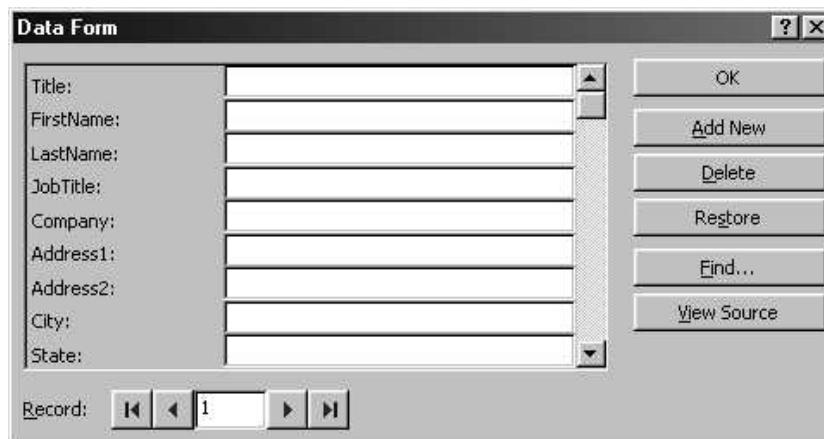


Figura 2.21. Fereastra de introducere/modificare date DATA FORM

Editarea sursei de date este operația prin care se introduc/modifică informațiile. Această operație se face prin intermediul unei ferestre specifice de tipul formă de introducere în bazele de date, fereastra DATA FORM (figura 2.21)

Datele pentru fiecare câmp se introduc în caseta corespunzătoare identificării câmpului. În subsolul ferestrei este marcată înregistrarea accesată curent, deplasarea la cea dorită putându-se asigura prin butoanele marcate cu săgeți. Eliminarea unei înregistrări se face prin butonul DELETE, iar adăugarea uneia noi, prin ADD NEW. Lucrul asupra sursei de date se consideră terminat prin acționarea butonului OK.

Sistemul de cereri existent în procesul de interconectare a documentelor consideră aprioric toate articolele din sursa de date, în ordinea introducerii lor. Utilizatorul are posibilitatea să introducă atât criterii de selectare a unor anumite înregistrări, cât și de ordonare a documentelor finale obținute prin interconectare. Sistemul de cereri este asigurat prin fereastra QUERY OPTIONS, prin paginile: FILTER RECORDS (pentru specificarea criteriilor de selecție) și SORT RECORDS (pentru ordonare).

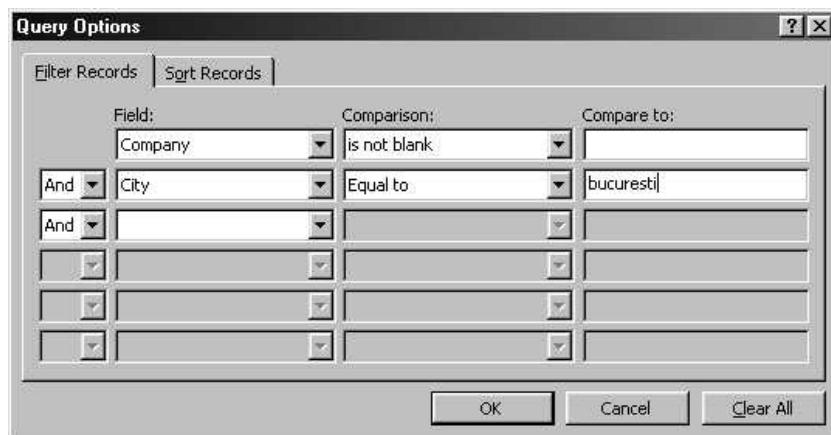


Figura 2.22. Exemplu de criteriu de selecție,
pagina FILTER RECORDS, fereastra QUERY OPTION

Alegerea modului de ordonare se realizează pe principii similare ordonării în tabele, prin maxim trei nivele de specificare, pentru fiecare dându-se identificarea câmpului și ordinea dorită.

Criteriile de selecție sunt date ca expresii logice construite pe baza unor relații între valorile dintr-un câmp identificat și valori constante. Un exemplu pentru fereastra QUERY OPTION cu indicarea unui criteriu de selecție este dat în figura 2.22.

În acest exemplu am considerat că se utilizează din sursa de date doar înregistrările pentru care câmpul COMPANY este completat efectiv (prima relație dată – company IS NOT BLANK) și (AND) pentru care în câmpul „city” valoarea este „bucuresti” (a doua relație – city EQUAL To bucuresti).

Existența documentului primar și a sursei de date cu informații efective face posibilă finalizarea lucrului cu sistemul de interconectare a documentelor. Anterior obținerii rezultatului final, există și posibilitatea realizării unei verificări pentru detectarea eventualelor erori de definire. Acest lucru se realizează prin intermediul shortcut-ului CHECK FOR ERRORS. Rezultatul final se obține prin aplicarea butonului MERGE, din zona 3 a ferestrei de asistență sau prin shortcut-ul MAIL MERGE, din bara specifică acestei activități.

Deoarece majoritatea rezultatelor finale sunt orientate fie spre imprimare directă, fie indirectă, prin reținerea într-un fișier multidocument, pentru a evita dialogul suplimentar, se pot folosi unul din shortcut-urile specifice, MERGE TO NEW DOCUMENT și MERGE TO PRINTER. Pentru aceste variante de finalizare, fiecare document final se începe cu o pagină nouă.



Figura 2.23. Fereastra MERGE

În varianta generală de cerere a realizării interconectării, specificațiile se dău prin fereastra MERGE (figura 2.23.). Această fereastră permite în plus indicarea domeniului de înregistrări la care se aplică considerarea selecției.



Figura 2.24. Fereastra MERGE TO SETUP

Varianta de ieșire neaccesată direct este transmisia rezultatului în sistemul de e-mail (ELECTRONIC MAIL). Pentru această modalitate de finalizare, fiecare document final va reprezenta câte o transmisie e-mail. Pentru acest serviciu, utilizatorul trebuie să stabilească și o serie de elemente caracteristice comunicării prin e-mail, lucru posibil prin actionarea butonului SETUP din fereastra MERGE.

Elementele, obligatorii în serviciul e-mail, sunt adresa electronică de destinație a mesajului și rezumarea subiectului mesajului. Indicarea acestor caracteristici se face prin fereastra MERGE TO SETUP (figura 2.24.) și împun existența în sursa de date a unui câmp special care să conțină adresa electronică pentru destinatarul fiecărui mesaj. Comunicarea electronică se face prin apelarea la o altă componentă a pachetului de aplicații MSOffice, programul MICROSOFT OUTLOOK.

III. CALCUL TABELAR

III.1. Prezentare generală

Programul de calcul tabelar din cadrul pachetului MSOffice poartă numele de Excel. Ca toate programele similare, el are o serie de facilități generale și care îl face util în activitatea de birou, ca instrument atât de reținere a informațiilor, cât și de realizare a unor calcule.

Făcând apel la definiția pentru baze de date, se observă că instrumentele cu care se lucrează sunt similare acestora, prin înregistrarea datelor utile împreună cu procesele de prelucrare a lor.

Aplicația Excel asigură următoarele facilități importante:

- înregistrarea datelor prin intermediul tabelelor performante;
- sistem foarte simplu de identificare a datelor;
- segment de realizare a calculelor înzestrat cu o bibliotecă de procese de calcul predefinite;
- realizarea independentă și specifică a operațiilor de editare în care se ține cont și de asocierea proceselor de calcul;
- implementarea internă a unui sistem asistat de specificare a prezentărilor vizuale grafice.

III.2. Formate de lucru și identificarea datelor

Instrumentul de memorare și reținere externă a informațiilor în sistemul de calcul tabelar, aflat în corespondență cu noțiunea externă de fișier, este lucrarea. Ea are aceeași identificare cu a fișierului în care este înregistrată extern, dacă acest lucru a fost realizat, sau numele implicit BOOK, urmat de o valoare numerică, dacă lucrarea nu a fost salvată sau a fost înregistrată extern, fără specificarea unui alt nume.

O lucrare poate fi structurată în unul sau mai multe segmente tabelare, numite fișe de calcul (WORKSHEET). Acestea se găsesc în aceeași fereastră de lucru din aplicația EXCEL, suprapuse una peste alta și identificate prin intermediul unei cleme de acces. Numele unei fișe de calcul poate fi cel implicit, format din SHEET și un număr de ordine, sau cel specificat de utilizator, prin înlocuirea numelui implicit în clema de acces (comanda SHEET din meniul FORMAT, subactivitatea RENAME). La inițializarea unei lucrări noi, aceasta este generată cu trei file de calcul. Utilizatorul are posibilitatea ca în orice moment să introducă în lucrare fișe de calcul noi (comanda WORKSHEET din meniul INSERT) sau să eliminate fișa de calcul activă curent (comanda DELETE SHEET din meniul EDIT). În figura 3.1. dăm un exemplu de lucrare inițială (BOOK) cu fișă a două, cu numele definit utilizator („calcule”).

Fișele de calcul se găsesc, în mod normal, în starea vizibilă. Această stare poate fi schimbată în starea ascunsă prin comanda SHEET din meniul format, subactivitatea HIDE, schimbarea fiind relativă la fișa de calcul considerată activă. Revenirea la starea vizibilă se face prin aceeași comandă, subactivitatea UNHIDE, care permite alegerea fișei de calcul care va deveni vizibilă prin intermediul unei ferestre de listare a tuturor fișelor de calcul ascunse (figura 3.2.).

O fișă de calcul este împărțită printr-o grilă în celule, fiecare fiind o entitate separată, cu proprietăți și conținut individuale. Proprietățile pentru fiecare celulă se manifestă prin proprietățile de scriere în celulă, aceleași cu proprietățile de scriere

prezentate la aplicația WORD. La ele se adaugă caracterizarea conținutului celulei ca tip de informație vehiculată. Din acest punct de vedere, celulele pot fi cu informație introdusă sau calculată.

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								

Figura 3.1. Exemplu de lucare

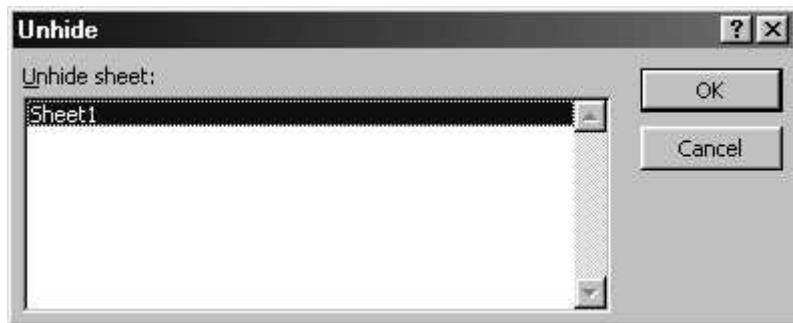


Figura 3.2. Fereastra UNHIDE

Pentru celulele cu informația introdusă, tipul conținutului este dependent de modalitatea de depunere a informației utile, iar pentru cele calculabile de tipul rezultatului aplicării formulelor de definire a prelucrării. La introducerea directă a valorii putem avea informație:

- de factură numerică, dată fie ca număr întreg, fie ca număr real, folosind ca separator zecimal punctul;
- de tip dată și timp, specificată printr-o formă specifică acestei categorii (vezi comenziile de dată și timp pentru sistemul de operare DOS);
- text, care nu este de un tip normal recunoscut sau începe prin intermediul unui apostrof.

Celulele se definesc automat drept calculabile, dacă primul caracter specificat la introducerea unei informații este unul din caracterele de operații plus („+”), minus („-“) sau semnul egal („=”). În acest caz, specificația cu care se continuă trebuie să fie o formulă corectă, din punct de vedere al construcției matematice.

Identificarea celulelor se face în funcție de locul de utilizare al identificării. Pentru celulele identificate în aceeași fișă de calcul, pentru identificarea unei celule unice se folosește concatenarea între identificarea literară a coloanei și identificarea numerică a liniei în care se găsește celula. Astfel, specificația „d8” indică celula din coloana a patra (coloana d) și linia a opta.

Pentru realizarea unor operații complexe, sau ca argument pentru o serie de funcții calculabile, se poate realiza o identificare a unei grupări de celule reprezentată ca o zonă dreptunghiulară continuă. Specificarea unei astfel de zone se realizează printr-o construcție în care apar două identificări de celule separate prin caracterul două puncte („..”), prima reprezentând colțul stânga sus al zonei, iar cel de-al doilea, colțul dreapta jos. Astfel, o construcție „c3:f5” desemnează o zonă dreptunghiulară cu vârfurile în celulele c3, c5, f5, f3, deci formată cu 4 coloane și trei linii (în total 12 celule).

Dacă identificarea se realizează în fișe de calcul diferite din aceeași lucrare, la modul de identificare dat mai sus se adaugă, în față, numele fișei de calcul și separatorul semn de exclamare („!”).

Pentru cazul în care este necesară specificarea unei celule sau grup de celule din altă lucrare, se realizează o identificare completă, printr-o construcție formată din identificarea fișierului în care este memorată lucrarea, identificarea fișei de calcul și specificația celulei sau grupului de celule (ultimele două indicate mai sus).

Identificarea fișierului se realizează prin cuprinderea între apos-trofuri a căii de directoare pentru regăsirea fișierului (dacă este cazul), urmată de cuprinderea între paranteze drepte a numărului fișierului, completat cu tipul fișierului (”.XLS”)

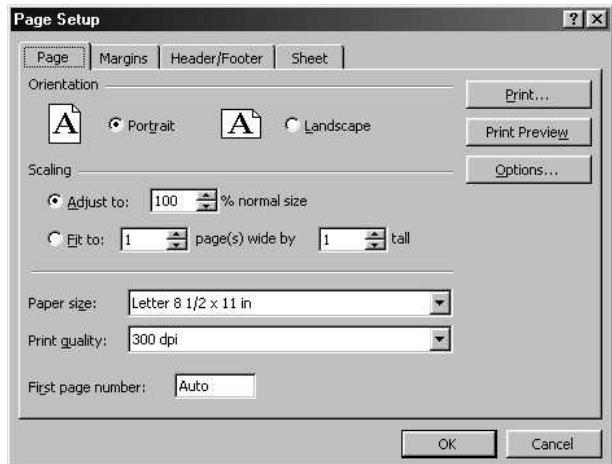
III.3. Caracteristici de imprimare

Pentru obținerea externă a rezultatelor, este necesară stabilirea caracteristicilor suportului extern (hârtiei). Acest lucru se realizează, pentru fișa de calcul curentă, prin intermediul comenzi PAGE SETUP din meniul file. Același lucru se poate obține și în momentul previzualizării ieșirii unei fișe de calcul, prin utilizarea meniului cu același nume din fereastra de vizualizare.

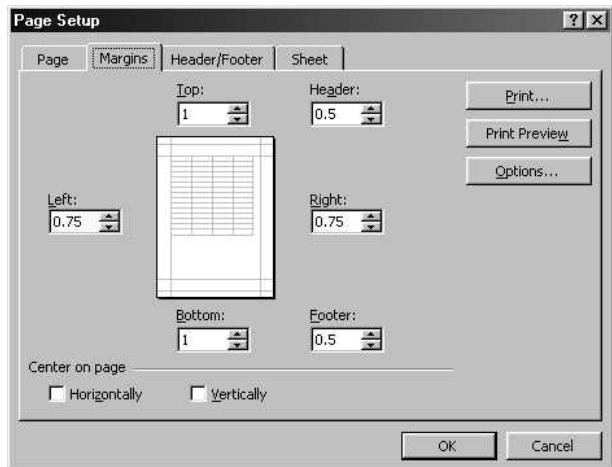
Fereastra de stabilire a caracteristicilor (figura 3.3.) conține mai multe pagini de specificații, și anume:

- PAGE, pentru indicarea orientării paginii fizice de hârtie, un procent de conversie al ieșirii (sub 100 % - micșorarea imaginii, peste 100 % - mărirea imaginii), dimensiunea tipografică a paginii fizice de hârtie, calitatea utilizată pentru imprimare și valoarea de început pentru numerotarea paginilor;
- MARGINS, pentru dimensionarea marginilor care vor fi utilizate în obținerea paginii imprimate din pagina fizică și posibilele centrări în pagină ale părților utile;
- HEADER/FOOTER, folosite la construcția antetelor și subsolurilor fixe din paginile imprimate;
- SHEET, prin care se indică zona imprimabilă din fișa de calcul, zonele de regăsire a titlurilor, ordinea de considerare a segmentelor de acoperire a zonei imprimabile (segmente verticale subîmpărțite – DOWN, THEN OVER, respectiv orizontale – OVER, THEN DOWN).

Facem observația că, pentru zona imprimabilă dintr-o fișă de calcul, există comanda directă PRINT AREA în meniul FILE cu subactivitățile SET PRINT AREA pentru stabilire și CLEAR PRINT AREA, pentru anularea indicației valabile. La aplicarea subactivității SET, este necesar ca zona respectivă să fie selectată.



a. Pagina PAGE



b. Pagina MARGINS

Figura 3.3. Fereastra PAGE SETUP

Antetele și notele de subsol care pot fi introduse în paginile imprimante pot fi constituite pe baza unor formate standardizate, selectable din pagina specifică a ferestrei PAGE SETUP. În afara construcțiilor predefinite pentru ambele elemente, utilizatorul își poate construi propriile formate pe baza butoanelor CUSTOM HEADER și CUSTOM FOOTER.

Definirea efectivă a acestor elemente se realizează prin ferestre specifice HEADER, respectiv FOOTER. Fiecare zonă este împărțită în trei segmente, câte unul pentru zonele stângă, centrală și dreaptă. În orice zonă se pot introduce texte fixe și texte variabile preluate din sistem (date curente, numerotarea paginilor, momentul curent al zilei, care se pot introduce prin butoanele existente în fereastră – figura 3.4.).

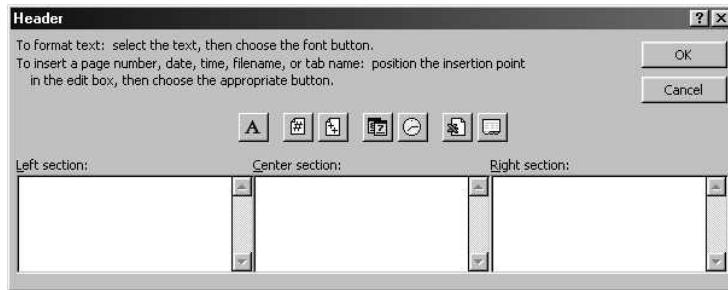


Figura 3.4. Forma constructivă antet – notă subsol.

III.4. Operații de tip editare specifice

Operațiile de editare se realizează asupra unei celule, cea selectată curent, sau asupra unei zone dreptunghiulare selectată. Se pot aplica acțiuni de tip copiere sau mutare. Realizarea acestora este una specifică și se face printr-un CLIPBOARD intern, nu prin cel general al platformei WINDOWS. Zona supusă operației va fi numită zonă sursă, iar cea care va rezulta va fi numită zonă destinație. Transferul și modul de lucru țin cont inclusiv de existența formulelor de calcul pentru celulele din zona destinației.

Mutarea informațiilor se realizează prin cuplu de comenzi CUT-PASTE, iar mutarea prin cuplul COPY-PASTE. Comanda PASTE este una implicită, deoarece după inițierea mutării/copierii prin CUT/COPY și pozitionarea pe destinație este suficientă apăsarea tastei „ENTER”, pentru realizarea acțiunii PASTE.

Regula care se aplică la mutare relativ la zonele implicate este că ele au exact aceeași dimensiune. Din această cauză, pentru destinație este suficientă indicarea prin punctare doar a colțului stânga sus a destinației. Aplicarea pentru formule în sursă se face fără modificarea acestora (netranslatarea formulelor).

Copierea funcționează diferit, în funcție de dimensiunile zonei sursă, astfel:

1. Sursa este formată dintr-o singură celulă; destinația poate fi de orice dimensiune, informația din sursă sau formula de definire a calculelor din sursă se copiază în toate celulele zonei destinație;

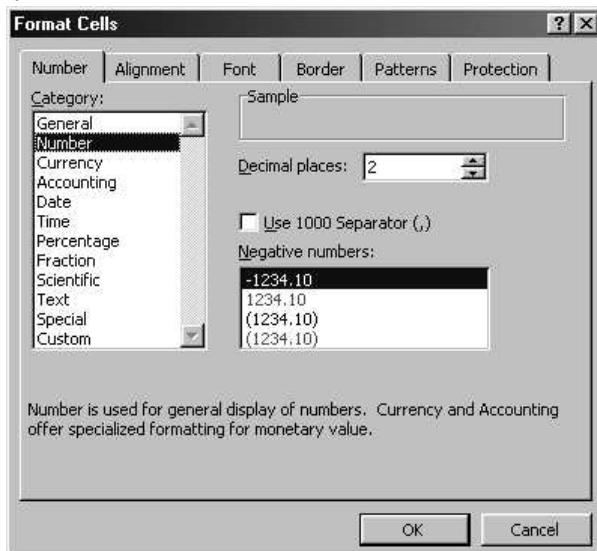
2. Sursa este o zonă formată dintr-un număr n de celule aflate pe o singură linie; destinația poate fi orice zonă dreptunghiulară cu număr n de coloane, selectată efectiv, integral sau prin celule aflate pe aceeași coloană, sau o zonă identică dimensional, selectată complet sau dată prin celula aflată cel mai la stânga; cele n celule sursă se copiază pe fiecare linie a destinației;

3. Sursa este o zonă formată dintr-un număr m de celule aflate pe o singură coloană; copierea funcționează similar cazului 2, înlocuind coloanele cu liniile;

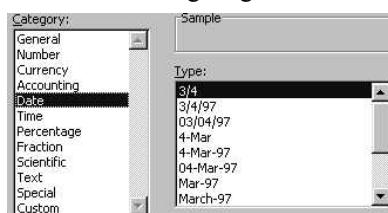
4. Sursa este o zonă dreptunghiulară efectivă; destinația este de aceeași dimensiune; se realizează o copiere unu la unu a celulelor sursei în destinație.

Specific pentru operația de copiere este translatarea formulelor care intervin în zona sursă. Prin această operație se modifică automat identificările celulelor care intervin în formule, indiferent de fișa de calcul și lucrarea în care ele se regăsesc. Modificarea identificărilor se face prin translație, pe temeiul unui vector definit pe baza colțurilor stânga sus ale sursei și destinației care dă deplasarea verticală și orizontală aplicată specificațiilor celulelor. Astfel, dacă diferența dintre colțurile stânga sus ale sursei și destinației este de 5 coloane la dreapta și 6 linii în jos, atunci fiecare identificare de celulă care intervine într-o formulă are specificarea de coloană deplasată cu 5 coloane de dreapta și numărul de linie crescut cu 6.

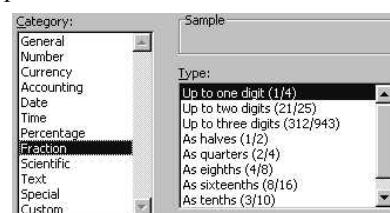
Tot ca operații de editare pot fi considerate și activitățile de ștergere și inserare care se pot aplica la nivelul linilor și/sau coloanelor. Prin poziționarea într-o celulă și utilizarea comenzii DELETE din meniu EDIT se deschide fereastra de ștergere din care se poate selecta ștergerea fie a întregii linii, fie a întregii coloane. Operațiile de inserare de linii coloane se realizează prin comenziile ROWS/COLUMNS din meniu INSERT. Linia/coloana introdusă suplimentar în fișă de calcul se va găsi înaintea celei pe care se găsește curent cursorul, producând, în același timp, și renumerotarea linilor/coloanelor.



a. Pagină generală – exemplificare Format numeric



b. Exemplificări data curentă



c. Exemplificări Format fracție

Figura 3.5. Pagina NUMBER – Fereastra FORMAT CELLS

III.5. Formatarea fișelor de calcul

Lucrul într-o fișă de calcul se poate face doar la nivelul tabelului predefinit general sau ținând cont și de ieșirile externe posibile, pe baza unor zone identificate special drept tabele definite utilizator. Pentru astfel de definiții, putem alege inclusiv forme speciale definite pentru caracteristicile celulelor. Stabilirea acestor elemente se face prin intermediul comenzi CELLS din meniu FORMAT

Accesul la această activitate produce deschiderea unei ferestre de comunicare, cu mai multe pagini de selecții de caracteristici posibile.

Tipul informației memorate

Pagina NUMBER a fereastrăi FORMAT CELLS (figura 3.5.) permite specificarea tipului informației care se va introduce în celula/celulele selectate. Această fereastră conține toate formatele acceptate pentru datele înregistrate în celule, cu toate variantele în care datele se pot specifica/vizualiza. Figura 3.5. prezintă variante exemplificative pentru formatele selectable.

Alinierea informației

Pagina ALIGNMENT din fereastra FORMAT CELLS (figura 3.6.) are rolul de a permite specificarea modalității de aliniere orizontală și verticală a datelor în interiorul celulelor selectate. Alinierea orizontală este cea utilizată curent și în operațiile de editare, permitând alinierea la stânga, centrată și la dreapta și scrierea cu aliniere dublă (stânga și dreapta cu modificarea dimensionării spațiilor). Alinierea verticală este similară celei orizontale, fiind permise aliniere sus, centrată, jos și dublă (aceasta în sensul sus și jos, cu modificarea spațiilor dintre linii).

Tot în această pagină se indică modul de orientare al informației introduse sau obținute prin calcul. Acest element este dat prin unghiul exprimat în grade pe care textul îl face cu orizontală, scrierea normală având această caracteristică 0, scrierea verticală de jos în sus are caracteristica 90, iar scrierea verticală de sus în jos are caracteristica -90.

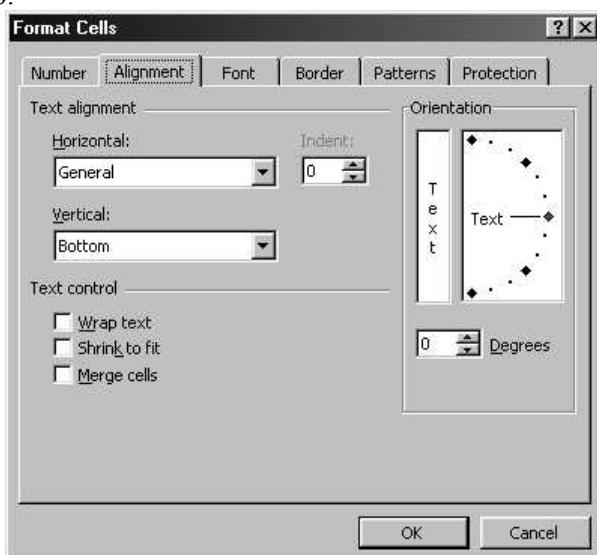


Figura 3.6. Fereastra FORMAT CELLS – Pagina ALIGNMENT

În scrierea textelor în celule există posibilitatea ca imaginea textului să depășească dimensiunea unei celule. Utilizatorul poate indica în acest caz împărțirea textului pe mai multe linii, prin selectarea casetei WRAP TEXT.

O altă posibilitate este cea prin care mai multe celule să fie reunite pentru a conține o singură informație. Pentru acest caz, la trecerea la formatul celulelor trebuie selectate toate celulele implicate și în pagina ALIGNMENT să se aleagă selecția MERGE CELLS.

Chenarul celulelor

Pentru o celulă sau o zonă dreptunghiulară de celule selectate, se pot alege modalitățile de liniere a marginilor celulei sau a zonei și celulelor componente. Pentru realizarea acestei operații, se folosește pagina BORDER din fereastra FORMAT CELLS (figura 3.7.). Zona dreaptă conține exemplificarea alegерilor de liniere cu butoane pentru fiecare liniere posibilă. Același efect cu utilizarea butoanelor îl are un click, cu cursorul mouse poziționat în zona de liniere din fereastra exemplificativă.

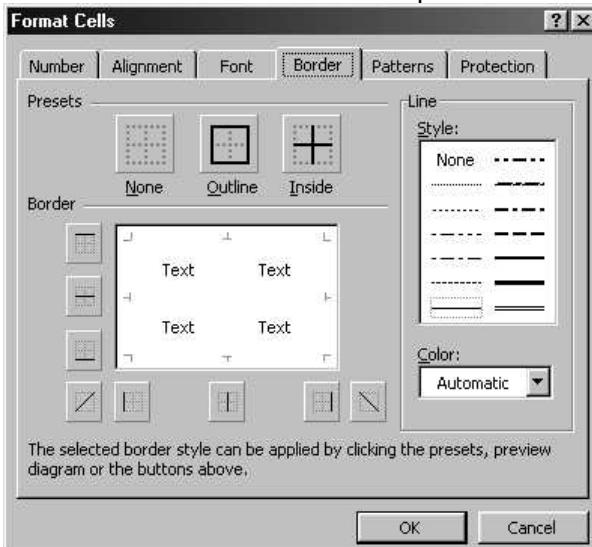


Figura 3.7. Fereastra FORMAT CELLS – Pagina BORDER

În zona dreaptă a paginii, utilizatorul poate să aleagă tipul și grosimea liniei utilizată pentru marginile trasate, ca și culoarea utilizată pentru liniere.

În afara caracteristicilor specificate explicit în subcapitolele de mai sus, prin pagini separate ale ferestrei FORMAT CELLS se pot stabili – carceristicile de scriere efectivă a textului (specificațiile de fontă – pagina FONT), culoarea de fond a celulelor (pagina PATTERN) și protecția aplicată pentru celule (blocate și/sau ascunse – pagina PROTECTION).

III.6. Specificarea formulelor de calcul

Specificarea formulelor de calcul se realizează direct prin introducerea curentă într-o celulă. Identificarea unei celule calculabile se face prin existența ca prim caracter introdus a simbolului egal („=”), urmat de o formulă de calcul corectă.

Relațiile de calcul pot fi identificare de celulă, funcție sau o construcție matematică realizată pe baza operațiilor elementare de calcul și a operanzilor. Operanzzii pot fi, la rândul lor, funcții, identificări de celule, constante sau alte construcții matematice.

Pentru a ușura specificarea operanzilor care sunt identificatori de celule, după introducerea unui caracter corespunzător unei operații matematice elementare, putem

puncta direct o celulă, care va fi trecută în chenar cu linie punctată, identificarea sa fiind trecută în formula introdusă curent.

O dată cu introducerea caracterului egal, ca prim caracter, bara de identificare aflată sub barele de meniu se modifică, în sensul că primul element, inițial identificând celula în care se găsește cursorul, trece în identifier/selector funcțional. Funcția vizibilă este ultima funcție utilizată. Săgeata de trecere la zona selectivă conține cele mai recente 10 funcții folosite și sintagma „MORE FUNCTION”, pentru accesul la fereastra de alegere a funcției dorite din cele disponibile. Accesul la funcții se poate face și prin comanda FUNCTION din meniul INSERT sau prin butonul de shortcut, cu identificarea PASTE FUNCTION.

Fereastra de alegere a funcției dorite, cu numele PASTE FUNCTION (figura 3.8.) conține trei elemente: gruparea funcțiilor pe categorii, lista funcțiilor din categoria selectată și descrierea formatului și a utilizării funcției selectate. Primele două elemente sunt constituite în casete de listare.

Categoriile de împărțire a funcțiilor sunt:

- 1) funcții de natură economico-financiară (FINANCIAL);
- 2) procese de lucru cu valori de date și timp, sub diferite formate (DATE & TIME), inclusiv funcția de obținere a datei și momentului curent de timp, conform informației reținute în ceasul sistem;
- 3) realizarea calculelor matematice și trigonometrice uzuale (MATH & TRIG);
- 4) funcții legate de teoria probabilităților și de calculele de statistică matematică (STATISTICAL);
- 5) specificații pentru căutare, legături și referințe (LOOKUP & REFERENCE);
- 6) construcții prin care se lucrează cu fișele de calcul EXCEL, ca instrumente de baze de date (DATABASE);
- 7) facilități legate de prelucrarea sirului de caractere (textelor - TEXT);
- 8) funcții de factură logică, inclusiv pentru asigurarea posibilității calculelor condiționate (LOGICAL);
- 9) obținerea informațiilor relative la o referință, celulă, eventuala apariție a unei erori, într-un cuvânt, funcții informative (INFORMATION).

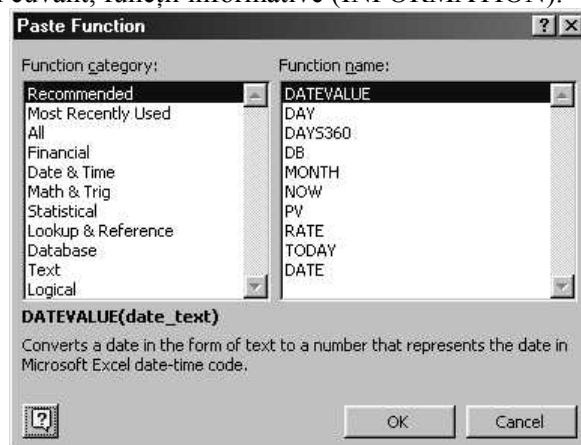


Figura 3.8. Fereastra PASTE FUNCTION

În afara acestor categorii, caseta de listare conține și două categorii speciale, una pentru cele mai recente 10 funcții utilizate (MOST RECENTLY USED) și selecția listării tuturor funcțiilor disponibile (ALL). De asemenea, după selectarea asistenței interne cu specificarea unei funcții, intervine în plus categoria prin care se dău funcțiile recomandabile relative la ajutorul solicitat (RECOMMENDED).

III.7. Realizarea graficelor

În practică, un mod de lucru semnificativ pentru studiile socio-economice este reprezentat de compararea rezultatelor activității sau studiului curent. Cuprinderea acestor comparații în tabele, prin valorile efective, nu dă însă o imagine semnificativă pentru evoluția temporală a unei activități economice sau pentru gradul de cuprindere a unui studiu. Sistemul de lucru cel mai semnificativ este cel în care informațiile sunt redată sub o formă grafică.

Problemele care intervin în astfel de reprezentări sunt legate, în special, de alegerea formatului de grafic care să corespundă cel mai bine tematicii studiate, să ofere o perspectivă cât mai coerentă a fenomenului care are loc.

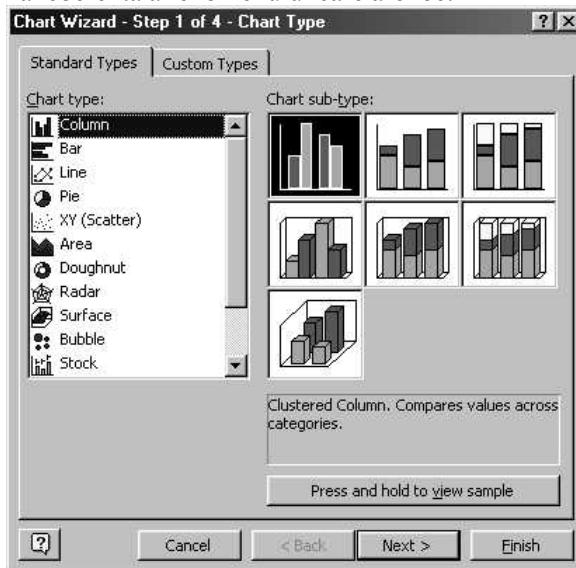


Figura 3.11. Fereastra CHART WIZARD – forma CHART STYLE

Acestea sunt motivele care au stat la baza cuprinderii în sistemele de birotică a unor procese automatizate de realizare a reprezentărilor grafice. Programele sau părțile de program specializate către o astfel de activitate au căptătat formate de comunicare externă cât mai simple și flexibile și care, cu aproximație, să răspundă pe deplin cerințelor majorității utilizatorilor. Un astfel de sistem de lucru, asistat integral prin forma cunoscută în literatura de informatică drept format WIZARD, se regăsește și în aplicația EXCEL.

Întreaga activitate se desfășoară prin intermediul unei ferestre multipagini evolutivă, care asigură comunicarea tuturor parametrilor grafici, de la alegerea formatului pentru ieșirea grafică inserată într-o fișă de calcul, până la inserția efectivă. Această fereastră de dialog permite ca în orice moment: să trecem de la etapa de specificări curente la cea anterioară sau ulterioară, să renunțăm la construcția grafică sau

să terminăm imediat cu implantarea graficului într-o fișă de calcul, considerând că toate elementele nespecificate au valorile implicate date de producătorul pachetului de birotică.

Trecerile indicate mai sus se realizează datorită prezenței în fiecare fereastră etapă a butoanelor NEXT (avans), BACK (revenire), CANCEL (renunțare) și FINISH (terminare).

Întreaga specificație grafică se realizează prin intermediul ferestrei CHART WIZARD în care, în bara de titlu se mai indică numărul pasului curent (STEP x OF 4, x este numărul pasului curent) și numele acestui pas.

Alegerea tipului de grafic

Stabilirea formei grafice se realizează ca primă etapă a construcției WIZARD (CHART STYLE figura 3.11). Selectarea formei celei mai semnificative trebuie să țină cont de aplicația pentru care s-a ales acest mod de prezentare, și depinde de experiența utilizatorului. Pentru a ușura alegerea, fereastra utilizată pentru primul pas al dialogului grafic oferă și o serie de explicații generale pentru fiecare tip de grafic.

Construcțiile posibile sunt împărțite în categorii, în funcție de forma imaginilor utilizate, formă legată și de folosirea lor în prezentări. Tipurile posibile sunt afișate în caseta de listare din partea stângă a ferestrei. Selectarea unei anumite categorii conduce la prezentarea, în dreapta ferestrei, a construcțiilor efective din cadrul tipului ales. Plasarea pe unul din formate are ca efect și o afișare generică a domeniului de aplicabilitate pentru graficul ales.

Indicarea seriilor de valori

Datele necesare realizării grafice depind de tipul construcției grafice alese în prima etapă. Specificarea datelor se face la nivelul ferestrei de forma CHART DATA SOURCE, din cadrul sistemului CHART WIZARD.

Pentru date există două modalități de prezentare: una globală și una element cu element. Prima este însă utilă doar atunci când datele se găsesc într-o singură fișă de calcul și ocupă o zonă continuă de celule. Fereastra de specificare a datelor este formată din câte o pagină pentru fiecare din aceste modalități de indicare o zonelor de valori.

Pagina DATA RANGE (figura 3.12.) permite indicarea globală a datelor și a modului de considerare a seriilor de valori utile. Pagina cuprinde în partea superioară o imagine a graficului care se obține, iar în partea inferioară caseta DATA RANGE, prin care se poate indica direct zona dreptunghiulară care conține informațiile utile pentru grafic. Această casetă, ca în cazul ferestrelor de construcție a formulelor conține un buton de trecere la fișele de calcul pentru selecția zonei de informații. Partea finală a ferestrei conține selectorul SERIES IN, pentru a indica considerarea seriilor de date pe linii (ROWS) sau pe coloane (COLUMNS).

Pagina SERIES are formatul general din figura 3.13. și permite specificarea individuală a seriilor de date. Ca și anterior, partea superioară a paginii prezintă o imagine a graficului care se obține pe baza datelor existente.

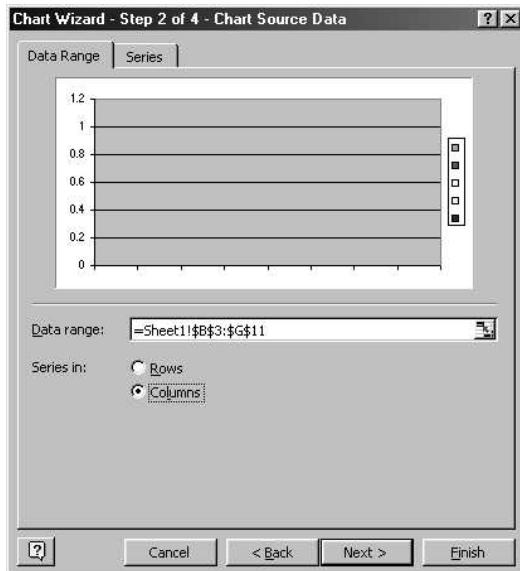


Figura 3.12. Fereastra CHART WIZARD,
forma CHART DATA SOURCE, pagina DATA RANGE

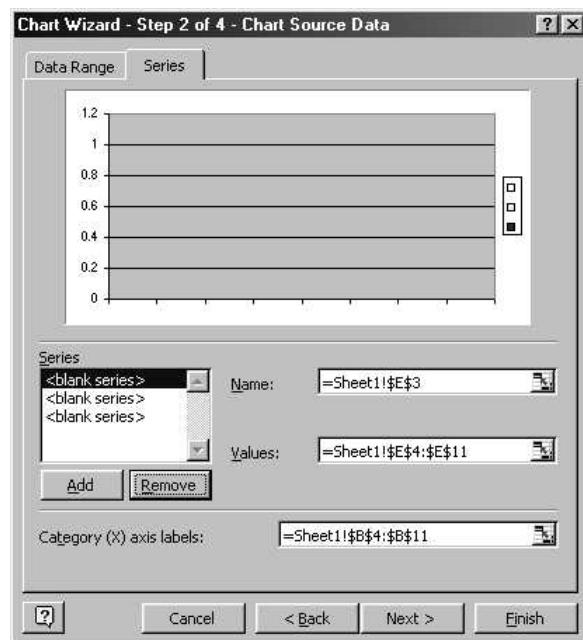


Figura 3.13. Fereastra CHART WIZARD,
forma CHART DATA SOURCE, pagina SERIES

Zona centrală este cea relativă la seriile de date utilizate. Acestea sunt listate în caseta SERIES, utilizatorul putând adăuga serii noi sau elimina seria selectată prin intermediul butoanelor ADD și, respectiv REMOVE. Pentru fiecare serie de date, prin casete de introducere separate, se pot indica numele seriei (caseta NAME) și domeniul de valori (caseta VALUES).

Zona finală oferă posibilitatea specificării valorilor utilizate pentru axa Ox a graficului (caseta CATEGOTY (X) AXIS LABELS).

Cele două pagini ale ferestrei cu forma CHART DATA SOURCE se află în interdependentă, specificațiile sau modificările din una din pagini producând modificarea automată a celeilalte. Modificările din pagina SERIES pot fi complexe, caz în care ele nu se vor reflecta și în pagina DATA RANGE. La introducerile din pagina DATA RANGE, zona de informații este văzută ca un tabel independent, configurat cu cap de tabel orizontal sau vertical, în funcție de modul de considerare a seriilor (pe linii, respectiv coloane), prima coloană, respectiv linie, fiind utilizată, implicit, pentru valorile care se vor trece pe axa Ox.

Stabilirea caracteristicilor grafice

Caracteristicile grafice se stabilesc în a treia etapă a construcțiilor grafice, prin intermediul formei CHART OPTIONS pentru fereastra CHART WIZARD. Caracteristicile sunt grupate pe pagini separate, fiecare prezentând și forma obținută pentru grafic cu datele și imaginea stabilite anterior, conform alegerilor curente pentru caracteristici.

Prima clasă de caracteristici, pagina TITLES (figura 3.14. a.), stabilește titlul general pentru grafic (caseta CHART TITLE) și denumirile asociate cu axele (caseta CATEGORY (X) AXIS pentru axa Ox și VALUE (Y) AXIS pentru axa Oy). Pentru unele tipuri de grafice pot exista și denumiri secundare pentru axe, acestea putând fi introduse, dacă este cazul, în casetele corespunzătoare. Dacă tipul de grafic nu poate conține denumiri secundare pentru axe, casetele destinate acestor valori sunt inactive.

A două pagină de caracteristici, AXES (figura 3.14. b.) stabilește modul de realizare a axelor graficului relativ la valorile scrise pe acestea. Marcarea casetelor CATEGORY (X) AXIS și/sau VALUE (Y) AXIS are ca efect marcarea și scrierea valorilor pe axa Ox și/sau respectiv, Oy. Dacă se scriu valori pentru axa Ox, atunci se poate face o selecție între utilizarea valorilor automate, pe categorii sau ca scală de timp.

Pagina GRIDLINES (figura 3.14. c.) permite specificarea trasării unor linii suplimentare în grafic, formând o grilă care ușurează înțelegerea imaginilor, fără a transcrie și valori efective în grafic. Liniiile se pot indica separat pentru fiecare din axe (trasare verticală sau orizontală) în două forme de trasare: trasarea pentru marcajele axelor (MAJOR GRID-LINES) sau pe subdiviziuni (MINOR GRIDLINES).

Pagina LEGEND (figura 3.14. d.) se folosește pentru a indica prezența în grafic sau nu a unei legende explicative. La alegerea prezenței acestei legende, utilizatorul poate indica modul de plasare a acesteia în spațiul grafic.

Paginile DATA LABELS (figura 3.14. e.) și DATA TABLE (figura 3.14 f) ajută la specificarea cuprinderii în grafic a etichetării cu valori sau cu marcaje a imaginilor și, respectiv, plasarea în spațiul de lucru grafic a unui tabel cu datele utile folosite efectiv.

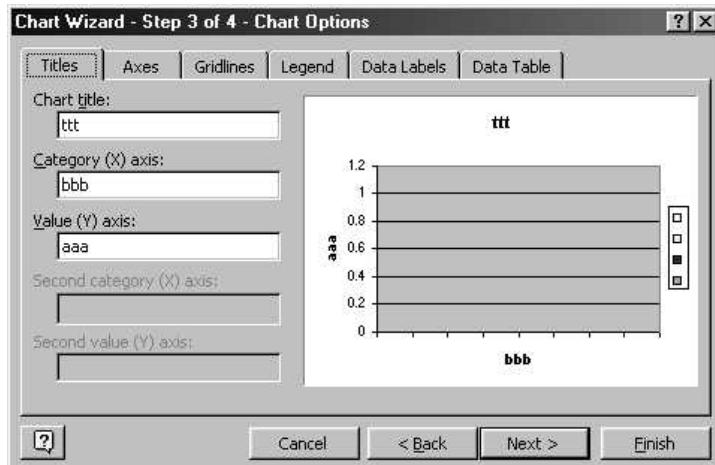


Figura 3.14. a. Fereastra CHART WIZARD, forma CHART OPTIONS, pagina TITLES (integral)

Primary axis <input checked="" type="checkbox"/> Category (X) axis <input checked="" type="radio"/> Automatic <input type="radio"/> Category <input type="radio"/> Time-scale <input checked="" type="checkbox"/> Value (Y) axis	Category (X) axis <input type="checkbox"/> Major gridlines <input type="checkbox"/> Minor gridlines Value (Y) axis <input checked="" type="checkbox"/> Major gridlines <input type="checkbox"/> Minor gridlines
b. Pagina AXES	
c. Pagina GRIDLINES	
<input checked="" type="checkbox"/> Show legend Placement: <input type="radio"/> Bottom <input type="radio"/> Corner <input type="radio"/> Top <input checked="" type="radio"/> Right <input type="radio"/> Left	Data labels <input checked="" type="radio"/> None <input type="radio"/> Show value <input type="radio"/> Show percent <input type="radio"/> Show label <input type="radio"/> Show label and percent <input type="radio"/> Show bubble sizes <input type="checkbox"/> Legend key next to label
d. Pagina LEGEND	
e. Pagina DATA LABELS	
f. Pagina DATA TABLE	

Figura 3.14. b-f. Fereastra CHART WIZARD, forma CHART OPTIONS (fără imagine grafică obținută)

Inserarea graficului în fișa de calcul

Utilizarea sistemului de grafică WIZARD permite atât conceperea unui grafic nou, cât și modificarea unui grafic existent, selectat la inițierea comenzii CHART din meniul INSERT.

Ultima etapă din realizarea grafică se referă la locul depunerii imaginii grafice care se obține în cazul generării unui grafic nou. La modificarea unui grafic realizat există încă plus posibilitatea mutării imaginii modificate din fișă de calcul în care exista inițial în altă fișă.

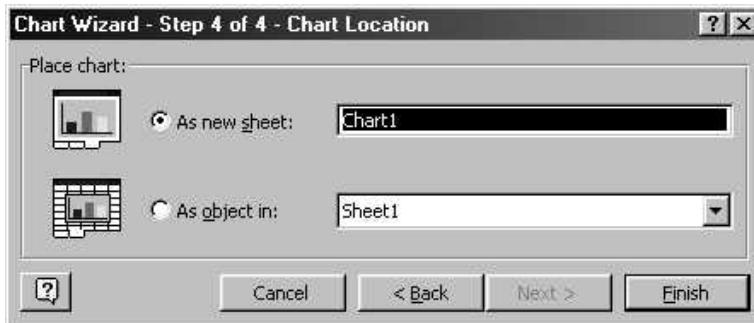


Figura 3.15. Fereastra CHART WIZARD – forma CHART LOCATION

Locul de plasare a graficului poate fi ales de utilizator, dintre fișele de calcul existente, cu identificările selectable prin caseta AS OBJECT IN. De asemenea, graficul poate fi generat/mutat într-o fișă nouă, cu un nume stabilit de utilizator, folosind caseta AS NEW SHEET. În al doilea caz, nu mai putem vorbi de fișă de calcul, deoarece în această fișă nu mai avem tabelul reformat standard. Deoarece această fișă are ca unică utilizare prezentarea imaginii grafice, o vom numi „fișă grafică”.

Specificarea locului de depunere a graficului generat se face prin forma CHART LOCATION, pentru fereastra CHART WIZARD (figura 3.15.).

BIBLIOGRAFIE

1. Couter G., Marquis, A., *Inițiere în Microsoft Office 2000*, Editura ALL, București, 1999
2. Fusaru D., Mareș D., Mmareș V., *Birotică*, Editura Fundației România de Mâine, București, 2001
3. Mareș D., *Bazele informaticii*, Editura Fundației România de Mâine, București, 2000
4. Kovacs, S. Excel 97, *Ghid de utilizare*, Editura Albastră, Cluj-Napoca, ed. IV, 2000
5. Mareș D., Fusaru D., Mihai G., Bârză S., *Birotică, Necesități și instrumente specifice*, Editura Fundației România de Mâine, București, 2002
6. MICROSOFT, *Excel '97 pas cu pas*, Editura Teora, București, 1998
7. MICROSOFT, *Windows '98*, Editura Teora, București, 1998
8. Văduva, I., Barbu, Gh., *Bazele informaticii*, Editura Tehnică, București, 1997
9. Bârză S., *Bazele informaticii și noțiuni de birotică*, Editura Fundației România de Mâine, București, 2002