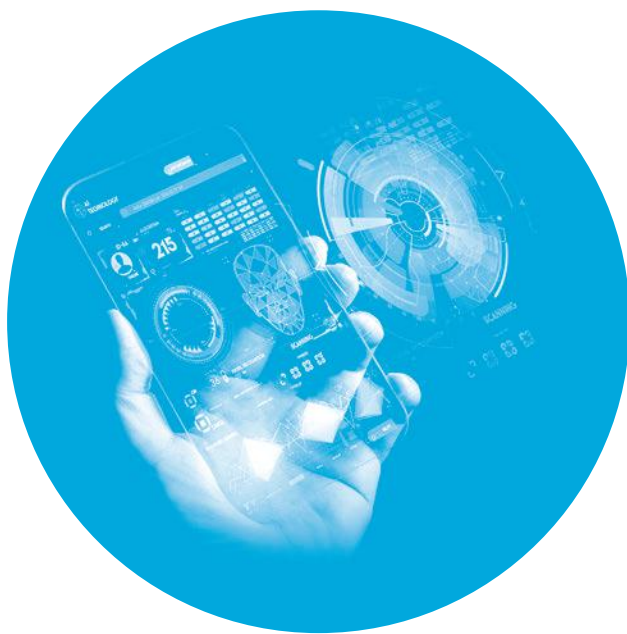


MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII

INFORMATICĂ

Manual pentru clasa a 11-a

ANATOL GREMALSCHI • IURIE MOCANU
LUDMILA GREMALSCHI



Știința, 2020

Acest manual este proprietatea Ministerului Educației, Culturii și Cercetării.

Manualul școlar a fost realizat în conformitate cu prevederile Curriculumului la disciplină, aprobat prin Ordinul Ministerului Educației, Culturii și Cercetării nr. 906 din 17 iulie 2019. Manualul a fost aprobat prin Ordinul Ministerului Educației, Culturii și Cercetării nr. 1219 din 6 noiembrie 2020, urmare a evaluării calității metodică-științifice.

Editat din sursele financiare ale *Fondului Special pentru Manuale*.

Comisia de evaluare:

Svetlana Brînză, profesor, grad didactic superior, Liceul Teoretic „N.M. Spătaru”, metodist DGETS, Chișinău (coordonator);

Gheorghe Chistruga, profesor, grad didactic superior, Liceul Teoretic „M. Eminescu”, Drochia;

Natalia Schițco, profesor, grad didactic superior, Liceul Teoretic „Socrate”, Chișinău;

Viola Bargan, profesor, grad didactic superior, Liceul Teoretic „Gh. Asachi”, Chișinău;

Ecaterina Adam, profesor, grad didactic unu, Liceul de Creativitate și Inventică „Prometeu-Prim”, Chișinău

Denumirea instituției de învățământ _____				
Manualul a fost folosit: _____				
Anul de folosire	Numele, prenumele elevului	Anul de studii	Aspectul manualului	
			la primire	la returnare

Dirigintele verifică dacă numele, prenumele elevului sunt scrise corect.

Elevii nu vor face niciun fel de însemnări în manual.

Aspectul manualului (la primire și la returnare) se va aprecia cu unul dintre următorii termeni: *nou, bun, satisfăcător, nesatisfăcător*.

Responsabil de ediție: Larisa Dohotaru

Redactor: Mariana Belenciuc

Corector: Maria Cornesco

Redactor tehnic: Nina Duduciuc

Machetare computerizată: Vitalie Ichim

Copertă: Romeo Șveț

ÎNȚEPRINDEREA
EDITORIAL-POLIGRAFICĂ

ȘTIINȚA

str. Academiei, nr. 3; MD-2028, Chișinău, Republica Moldova

tel.: (+373 22) 73-96-16; fax: (+373 22) 73-96-27

e-mail: prini_stiinta@yahoo.com

www.editurastiinta.md

Toate drepturile asupra acestei ediții aparțin Întreprinderii Editorial-Poligrafice Știința.

Descrierea CIP a Camerei Naționale a Cărții

Gremalschi, Anatol.

Informatică: Manual pentru clasa a 11-a / Anatol Gremalschi, Iurie Mocanu, Ludmila Gremalschi;

Comisia de evaluare: Svetlana Brînză (coordonator) [et al.]; Ministerul Educației, Culturii și Cercetării. –

Ch.: Î.E.P. Știința, 2020 (Combinatul Poligrafic). – 224 p.: fig., tab.

Proprietate a Min. Educației, Culturii și Cercet.

ISBN 978-9975-85-246-3

Imprimare la **COMBINATUL POLIGRAFIC**

str. Petru Movilă, 35, MD-2004, Chișinău, Republica Moldova, Comanda nr.

CUPRINS

Conținuturi	Umanist	Real	Pagina
Introducere			5
1. TIPURI DE DATE STRUCTURATE			
1.1. Date simple și date structurate	•	•	6
1.2. Tipuri de date <i>tablou</i>	•	•	10
1.3. Tipuri de date <i>șir de caractere</i>	•	•	26
1.4. Tipuri de date <i>articol</i>	•	•	40
1.5. Instrucțiunea with	•	•	49
1.6. Tipuri de date <i>mulțime</i>	•	•	52
1.7. Generalități despre fișiere	•	•	62
1.8. Fișiere secvențiale	•	•	71
1.9. Fișiere <i>text</i>	•	•	78
2. INFORMAȚIA			
2.1. Cantitatea de informație	•	•	92
2.2. Codificarea și decodificarea informației	•	•	95
2.3. Coduri frecvent utilizate	•	•	97
2.4. Informația mesajelor continue		•	103
2.5. Cuantizarea imaginilor		•	106
2.6. Reprezentarea și transmiterea informației		•	109
3. BAZELE ARITMETICE ALE TEHNICII DE CALCUL			
3.1. Sisteme de numerație		•	114
3.2. Conversiunea numerelor dintr-un sistem în altul		•	117
3.3. Conversiunea din binar în octal, hexazecimal și invers		•	119
3.4. Operații aritmetice în binar		•	122
3.5. Reprezentarea numerelor naturale în calculator		•	124
3.6. Reprezentarea numerelor întregi		•	125
3.7. Reprezentarea numerelor reale		•	128
4. ALGEBRA BOOLEANĂ			
4.1. Variabile și expresii logice		•	133
4.2. Funcții logice		•	137
4.3. Funcții logice frecvent utilizate		•	139

Conținuturi	Umanist	Real	Pagina
5. CIRCUITE LOGICE			
5.1. Circuite logice elementare		•	142
5.2. Clasificarea circuitelor logice		•	147
5.3. Sumatorul		•	147
5.4. Circuite combinaționale frecvent utilizate		•	151
5.5. Bistabilul <i>RS</i>		•	154
5.6. Circuite secvențiale frecvent utilizate		•	157
5.7. Generatoare de impulsuri		•	160
6. STRUCTURA ȘI FUNCȚIONAREA CALCULATORULUI			
6.1. Schema funcțională a calculatorului	•	•	163
6.2. Formatul instrucțiunilor		•	165
6.3. Tipuri de instrucțiuni		•	168
6.4. Limbajul cod-calculator și limbajul de asamblare		•	170
6.5. Resursele tehnice și resursele programate ale calculatorului	•	•	172
6.6. Memorii externe pe benzi și discuri magnetice	•	•	174
6.7. Memorii externe pe discuri optice	•	•	178
6.8. Vizualizatorul și tastatura	•	•	182
6.9. Imprimantele	•	•	184
6.10. Clasificarea calculatoarelor	•	•	187
6.11. Microprocesorul		•	189
7. REȚELE DE CALCULATOARE			
7.1. Introducere în rețele	•	•	192
7.2. Tehnologii de cooperare în rețea	•	•	195
7.3. Topologia și arhitectura rețelelor		•	197
7.4. Rețeaua <i>Internet</i>	•	•	201
7.5. Servicii <i>Internet</i>	•	•	206
8. MODULE LA ALEGERE			
8.1. Introducere în rețele	•	•	211
8.2. Programarea vizuală	•	•	216
8.3. Limbaje de marcare a hipertextului	•	•	221

INTRODUCERE

Dragi prieteni,

Impresionantele realizări în domeniul informaticii, crearea supercalculatoarelor și a calculatoarelor personale, apariția cyberspațiului, a *Internetului* și a realității augmentate presupun o cunoaștere profundă a principiilor de funcționare și a structurii calculatoarelor moderne, a metodelor de programare a acestora. Indiferent de specificul viitoarei activități profesionale a fiecărui absolvent de liceu, aceste cunoștințe vor servi drept o călăuză demnă de toată încrederea într-o lume digitală aflată mereu în schimbare.

Dezvoltarea continuă a informaticii ca știință fundamentală și dezvoltarea fulminantă a tehnologiei informației și comunicațiilor ca știință aplicativă are drept efect apariția de noi și noi echipamente digitale care, practic, se reînnoiesc la fiecare doi-trei ani. „Invazia” gadgeturilor, diversitatea crescândă a produselor-program, apariția *Internetului Obiectelor* cer revizuirea metodelor de învățare a informaticii, reorientându-le de la formarea simplor dexterități de acționare a butoanelor, de clicare și glisare a obiectele de pe ecranele tactile, la formarea competențelor de a utiliza și crea echipamente și produse digitale noi, cum ar fi programe de aplicații, documente Web, servicii digitale online, opere de artă digitală.

În lumea modernă, cele mai impresionante inovații apar la intersecția diverselor domenii ale cunoașterii. Drept exemplu pot servi digitalizarea radioului și a televiziunii, a muzicii și a artelor vizuale, extinderea comerțului online, creșterea rolului rețelelor sociale, afirmarea instruirii asistate de calculator. Apariția și dezvoltarea inteligenței artificiale, care, conform previziunilor mai multor personalități ilustre ale zilelor de astăzi, va schimba în esență evoluția civilizației umane, presupune conjugarea eforturilor specialiștilor din diverse domenii de activitate, indiferent de natura acestora.

Prin urmare, indiferent de profilul de liceu pe care îl urmează elevii, umanist sau real, pentru a-și atinge scopurile de dezvoltare personală, inclusiv pentru o carieră profesională de succes și o viață împlinită, ei neapărat vor avea nevoie de acele competențe digitale, formarea și dezvoltarea cărora reprezintă scopul acestui manual. Mai exact, la finele clasei a 11-a elevii vor putea:

- să implementeze într-un limbaj de programare de nivel înalt algoritmi de prelucrare a datelor structurate;
- să utilizeze bazele teoriei informației, aritmeticii de calculator și a algebrei booleene pentru elaborarea modelelor informatice;
- să interpreteze rezultatele furnizate de programele de calculator elaborate;
- să identifice structura generală a sistemelor digitale utilizate, a principiilor de funcționare a sistemelor de transmitere, stocare și de prelucrare a informației.

Studierea unuia dintre modulele la alegere va contribui la valorificarea metodelor și instrumentelor specifice prelucrărilor digitale, la dezvoltarea gândirii și a creativității în integrarea achizițiilor informatice cu cele din alte domenii, la conștientizarea importanței respectării regulilor de securitate, ergonomice și etice în crearea și difuzarea produselor digitale.

În scopul autoevaluării, recomandăm elevilor să descarce de pe pagina web <http://www.ctice.gov.md> a Centrului Tehnologii Informaționale și Comunicaționale în Educație testele respective, să rezolve probele din componența acestora și să compare răspunsurile lor cu răspunsurile-etalon, care pot fi obținute de la profesorii de informatică.

*Vă dorim succese,
Autorii*

Capitolul 1

TIPURI DE DATE STRUCTURATE

1.1. Date simple și date structurate

Cunoaștem deja că informația ce urmează să fie supusă unei prelucrări este accesibilă calculatorului în formă de **date**. Datele sunt constituite din cifre, litere, semne, numere, șiruri de caractere ș.a.m.d.

Într-un limbaj cod-calculator datele sunt reprezentate prin secvențe de cifre binare. De exemplu, la nivelul procesorului numărul natural 1 039 se reprezintă în sistemul de numerație binar ca:

```
100000001111
```

Pentru a scuti utilizatorul de toate detaliile legate de reprezentarea internă a datelor, limbajele de programare utilizează diverse tipuri de date. Amintim că prin **tip de date** se înțelege o **mulțime de valori** și o **mulțime de operații** care pot fi efectuate cu valorile respective.

În clasele precedente ați studiat următoarele tipuri de date:

- **integer/int**, destinat prelucrării pe calculator a numerelor întregi;
- **real/float**, pentru prelucrarea numerelor reale;
- **boolean/bool**, utilizat în cazul prelucrării valorilor de adevăr;
- **char/char**, destinat reprezentării și prelucrării caracterelor;
- **enumerare (enum)**, care include o mulțime ordonată de valori specificate prin identificatori;
- **subdomeniu** (doar în limbajul PASCAL), care include o submulțime de valori ale unui tip **integer**, **boolean**, **char** sau **enumerare**.

Din perspectiva limbajelor de programare, numerele întregi, numerele reale, valorile de adevăr, caracterele, valorile ordonate specificate prin identificatori se numesc **date simple**, iar tipurile enumerate mai sus – **tipuri de date simple**.

În general, datele simple nu sunt suficiente pentru a reprezenta pe calculator și a prelucra în mod eficient informațiile din lumea înconjurătoare.

De exemplu, în cazul prelucrării textelor, operațiile de ștergere, copiere și mutare se fac nu doar la nivel de caractere (tipul de date simple **char**), ci și la nivel de cuvinte, propoziții, rânduri, paragrafe și chiar pagini întregi.

Evident, un cuvânt sau o propoziție pot fi reprezentate ca o succesiune de caractere, adică printr-o structură care este formată dintr-o înșiruire de date simple de tipul **char**. Pentru a descrie astfel de structuri, în limbajele de programare de nivel înalt se folosesc tipuri de date denumite sugestiv *șiruri de caractere*. În consecință, literele alfabetului latin A, a, B, b, C, c, ..., Z, z se reprezintă pe

calculator prin datele simple de tipul `char`, pe când cuvintele formate din ele, de exemplu, *Liceu*, *Patrie*, *Informatica* etc. – prin date structurate de tipul *șir de caractere*.

Un alt exemplu ce ilustrează necesitatea utilizării datelor structurate constă în prelucrarea informațiilor referitoare la consumul zilnic de energie electrică al unei gospodării. Acest consum, în kWh, poate fi reprezentat pe calculator printr-un număr real (tipul de date simple `real` sau `float`). Dacă ne dorim însă să analizăm variațiile consumului zilnic de energie electrică pe durata unei luni, va trebui să reprezentăm datele respective printr-un tabel unidimensional, format dintr-un singur rând:

Ziua lunii	1	2	3	...	31
Consumul zilnic, kWh	13,4	18,6	9,4	...	15,0

În prima celulă a acestui tabel se va conține consumul în kWh din prima zi a lunii; în celula a doua – consumul din ziua a doua a lunii ș.a.m.d., până la celula 31, în funcție de numărul de zile ale lunii în cauză.

În cazul analizei consumului zilnic de energie electrică pe parcursul unui an, va trebui să folosim un tabel format din 12 rânduri și 31 de coloane, adică un tabel bidimensional:

Ziua lunii	1	2	3	...	31
Ianuarie	13,4	18,6	9,4	...	15,0
Februarie	12,9	14,3	21,7	...	...
...	...	...	...	...	...
Decembrie	11,7	10,4	3,4	...	25,3

Primul rând al unui astfel de tabel va conține consumul pentru fiecare din zilele lunii ianuarie; rândul al doilea – consumul pentru fiecare din zilele lunii februarie ș.a.m.d., până la rândul ce corespunde lunii decembrie.

Se observă că tabelele de mai sus reprezintă date structurate, fiind formate din date simple de tipul `real/float`. De obicei, pe calculator tabelele respective se reprezintă cu ajutorul unor tipuri speciale de date structurate, denumite *tablouri*.

Accentuăm faptul că tablourile conțin doar datele din celule, fără denumirile de rânduri și de coloane. Pe calculator, aceste denumiri sunt reprezentate prin alte mărimi, denumite *indici*. Evident, în cazul tabelelor unidimensionale este necesar doar un singur indice, iar în cazul tabelelor bidimensionale – de doi.

De exemplu, în cazul tabloului unidimensional ce reprezintă pe calculator consumul zilnic de energie electrică pe parcursul unei luni, în calitate de indice se folosește o mărime de tip întreg, care poate lua doar valorile 1, 2, 3, ..., 31.

În cazul tabloului bidimensional ce reprezintă consumul zilnic de energie electrică pe parcursul unui an, sunt necesari doi indici, unul pentru rânduri și altul pentru coloane. În calitate de indice pentru rânduri se folosește o mărime de tip enumerare, care ia valorile *Ianuarie*, *Februarie*, *Martie*, ..., *Decembrie*. Pentru indicele de coloane se folosește o mărime de tip întreg care ia valorile 1, 2, 3, ... 31.

Datele formate prin agregarea (unirea într-un tot) a unor date simple se numesc *date structurate*. Tipurile de date utilizate pentru definirea unor astfel de date se numesc *tipuri de date structurate*.

În afară de tipurile de date structurate *șiruri de caractere* și *tablouri*, descrise în linii generale mai sus, limbajele de programare oferă informaticienilor posibilitatea să utilizeze și alte tipuri de date structurate, cele mai des utilizate fiind *articolele*, *mulțimile* și *fișierele*.

Întrebări și exerciții

- ① Explicați semnificația termenului *tip de date*. Dați exemple.
- ② Care este diferența dintre datele simple și datele structurate?
- ③ Dați exemple de tipuri de date simple și tipuri de date structurate.
- ④ OBSERVĂ! Determinați tipul datelor, simple sau structurate, din exemplele ce urmează:

a)

138

b)

3.14

c)

Munteanu Elena

d)

Clasa a 11-a

e)

12	6	231	5
----	---	-----	---

f)

32.51	149,28	318,56	20013.9	0.4536	721.3
-------	--------	--------	---------	--------	-------

g)

4	635	-8	+27
72	41	319	432
16	-20	45	1830

În cazul datelor structurate, indicați tipul datelor simple din componența acestora.

Modele de răspuns:

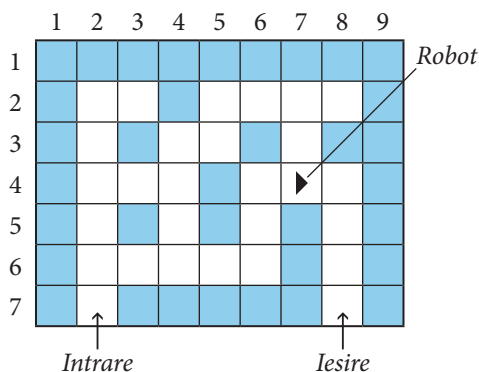
– Dată simplă, de tip *număr întreg*.

– Dată structurată de tip *tablou unidimensional*, format din numere întregi.

- ⑤ CREEAZĂ! Elaborați un tabel unidimensional ce conține plățile lunare pentru energia electrică folosită de familia voastră pe parcursul ultimului an calendaristic. Folosiți în acest scop informațiile din facturile de plată emise de compania ce vă furnizează energia electrică. Ce tipuri de date veți utiliza pentru reprezentarea pe calculator a informațiilor din acest tabel?
- ⑥ EXPLOREAZĂ! Site-ul Băncii Naționale a Moldovei conține informații referitoare la ratele oficiale medii de schimb (cursul valutar). De exemplu, în luna ianuarie a anului 2019, rata medie oficială de schimb pentru moneda unică a Uniunii Europene era: 1 Euro = 19,6501 Lei Moldovenesti. Elaborați:
 - un tabel unidimensional ce conține ratele lunare medii oficiale de schimb pentru ultimul an calendaristic;
 - un tabel bidimensional ce conține ratele lunare medii oficiale de schimb pentru ultimii trei ani.

Ce tipuri de date veți utiliza pentru reprezentarea pe calculator a informațiilor din aceste tabele?

- 7 CREEAZĂ! După o vizită la renumitele hrube de la Cricova, vizită pe durata căreia a avut posibilitatea să călătorească prin galeriile subterane ale acestui important obiect turistic, un informatician a construit un robot care găsește cel mai scurt drum de la o sală de expoziție la alta. În memoria calculatorului ce dirijează robotul, hrubele sunt reprezentate printr-un dreptunghi divizat în pătrățele (vezi figura de mai jos).



Pătrățelele hașurate reprezintă obstacolele (pereții galeriilor și sălilor de expoziție, standurile cu exponate), iar cele nehașurate – spațiile libere. Robotul poate executa doar instrucțiunile SUS, JOS, DREAPTA, STANGA, conform cărora se deplasează în unul din pătrățelele vecine. Dacă în acest pătrat este un obstacol, de exemplu, un perete sau un stand, robotul se oprește și nu mai reacționează la nicio instrucțiune.

Elaborați structura de date necesară pentru reprezentarea planului hrubelor de la Cricova pe calculator. Asigurați-vă că structura elaborată conține toate informațiile de pe plan: obstacolele, spațiile libere, pătrățelel în care se află robotul, intrarea, ieșirea.

- 8 ÎNVĂȚĂ SĂ ÎNVEȚI! Cunoașteți deja că datele structurate se obțin prin agregarea (unirea într-un tot) a unor date simple. Limbajele PASCAL și C++ permit agregarea nu doar a datelor simple, dar și a celor structurate. De exemplu, în celulele unui tabel pot fi memorate nu doar date simple (numere întregi, numere reale, caractere etc.), dar și date structurate, cum ar fi șirurile de caractere.

Se consideră lista elevilor unei clase de liceu:

Munteanu Ion
Florea Elena
...
Prisăcaru Alexandra

Elaborați structura de date necesară pentru reprezentarea unei astfel de liste pe calculator. Asigurați-vă că structura elaborată permite sortarea elevilor în ordine alfabetică.

1.2. Tipuri de date tablou

PASCAL

Pentru definirea tipurilor de date tablou, în limbajul PASCAL se utilizează cuvântul-cheie **array**. Tablourile sunt formate dintr-un număr fixat de componente de același tip, denumit tip **de bază**. Referirea componentelor se face cu ajutorul unui sau a mai multor **indici**.

Un tip de date *tablou unidimensional* se definește printr-o construcție de forma

type <Nume tip> = **array** [T_1] **of** T_2 ;

unde T_1 este tipul indicelui care trebuie să fie ordinal, iar T_2 este tipul componentelor (tipul de bază) care poate fi un tip oarecare.

Exemple:

- 1) **type** Vector = **array** [1..5] **of** real;
var x : Vector;_
- 2) **type** Zi = (L, Ma, Mi, J, V, S, D);
Venit = **array** [Zi] **of** real;
var v : Venit;
z : Zi;
- 3) **type** Ora = 0..23;
Grade = -40..40;
Temperatura = **array** [Ora] **of** Grade;
var t : Temperatura;
h : Ora;

Structura datelor din exemplele în studiu este prezentată în *figura 1.1*.

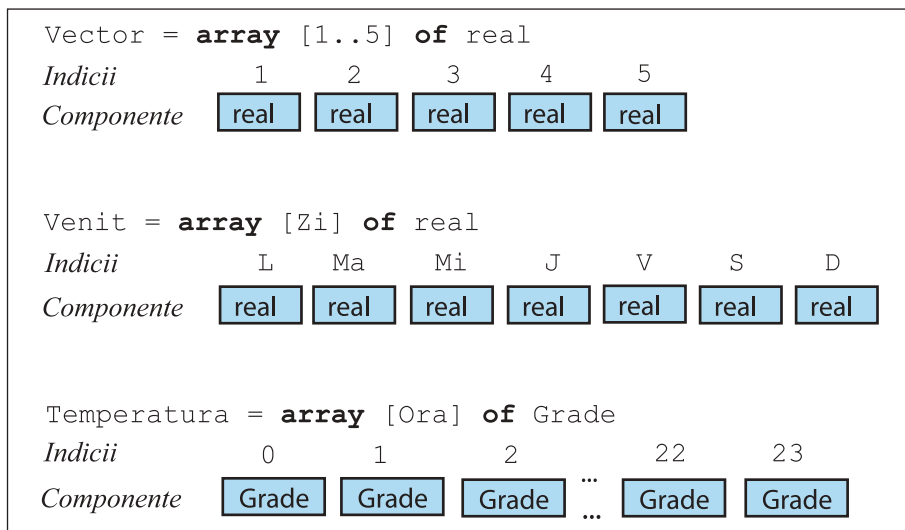


Fig. 1.1. Structura datelor de tip Vector, Venit și Temperatura

Fiecare componentă a unei variabile de tip *tablou unidimensional* poate fi specificată explicit, prin numele variabilei urmat de indicele respectiv încadrat de paranteze pătrate.

Exemple:

- 1) `x[1], x[4];`
- 2) `v[L], v[Ma], v[J];`
- 3) `t[0], t[15], t[23];`
- 4) `v[z], t[h].`

Asupra componentelor datelor de tip *tablou* se pot efectua toate operațiile admise de tipul de bază respectiv. Programul ce urmează afișează pe ecran suma componentelor variabilei `x` de tip **Vector**. Valorile componentelor `x[1]`, `x[2]`, ..., `x[5]` se citesc de la tastatură.

```
Program P78;
{ Suma componentelor variabilei x de tip Vector }
{ Numar fix de componente
type Vector = array [1..5] of real;
var x : Vector;
    i : integer;
    s : real;
begin
    writeln('Dati 5 numere:');
    for i:=1 to 5 do readln(x[i]);
    writeln('Ati introdus:');
    for i:=1 to 5 do writeln(x[i]);
    s:=0;
    for i:=1 to 5 do s:=s+x[i];
    writeln('Suma=', s);
    readln;
end.
```

Pentru a extinde aria de aplicare a unui program, se recomandă ca numărul de componente ale datelor de tip **array** să fie specificate prin constante.

De exemplu, programul P78 poate fi modificat pentru a însuma n numere reale, $n \leq 100$:

```
Program P79;
{ Suma componentelor variabilei x de tip Vector }
{ Numar flotant de componente
const nmax = 100;
type Vector = array [1..nmax] of real;
var x : Vector;
    n : 1..nmax;
```

```

    i : integer;
    s : real;
begin
    write('n='); readln(n);
    writeln('Dati ', n, ' numere:');
    for i:=1 to n do readln(x[i]);
    writeln('Ati introdus:');
    for i:=1 to n do writeln(x[i]);
    s:=0;
    for i:=1 to n do
        s:=s+x[i];
    writeln('Suma=', s);
    readln;
end.

```

În informatică, tablourile unidimensionale sunt utilizate foarte des pentru a sorta datele într-o anumită ordine, de exemplu, în cea crescătoare. Pentru o astfel de sortare se folosește următoarea metodă:

1) fiecare componentă a tabloului, începând cu prima, este comparată consecutiv cu fiecare dintre componentele ce urmează după ea;

2) dacă în urma comparării se constată că componenta curentă este mai mare decât cea cu care este comparată, ele sunt schimbate cu locul;

3) procesul de parcurgere a tabloului continuă până când vor fi comparate ultima și penultima dintre componente.

În limbajul cotidian al informaticienilor, o astfel de sortare se numește *sortare prin metoda bulelor*, întrucât componentele tabloului își schimbă pozițiile ca bulele dintr-un lichid: cele mai ușoare se ridică la suprafață, iar cele mai grele se duc la fund.

Pentru exemplificare, în programul ce urmează metoda bulelor este folosită pentru sortarea în ordine crescătoare a componentelor tabloului A. Componentele acestui tablou sunt citite de la tastatură, iar sortarea propriu-zisă se face în tabloul B.

```

Program P80;
{ Sortarea prin metoda bulelor }
const nmax=100;
type Tablou = array[1..nmax] of integer;
var A, B : Tablou;
    n, i, j : integer;
    x : integer;
begin
    write(,Dati numarul de componente n= ,);
    readln(n);
    writeln(,Dati componentele tabloului A:');
    for i:=1 to n do read(A[i]);
    readln;

```

```

B:=A;
for i:=1 to n-1 do
  for j:=i+1 to n do
    if (B[i]>B[j]) then
      begin
        x:=B[i];
        B[i]:=B[j];
        B[j]:=x;
      end;
writeln(,Tabloul initial A:');
for i:=1 to n do write(A[i], , );
writeln;
writeln(,Tabloul sortat B:');
for i:=1 to n do write(B[i], , );
writeln;
readln;
end.

```

Un tip de date *tablou bidimensional* se definește cu ajutorul construcției:

type <Nume tip> = **array** [T_1, T_2] **of** T_3 ;

unde T_1 și T_2 specifică tipul indicilor, iar T_3 – tipul componentelor.

Pentru exemplificare, în *figura 1.2* este prezentată structura datelor tipului Matrice:

type Matrice = **array** [1..3, 1..4] **of** real;

Matrice = array [1..3, 1..4] of real				
	1	2	3	4
1	real	real	real	real
2	real	real	real	real
3	real	real	real	real

Fig. 1.2. Structura datelor de tip Matrice

Componentele unei variabile de tip *tablou bidimensional* se specifică explicit prin numele variabilei urmat de indicii respectivi separați prin virgulă și încadrați de paranteze pătrate.

De exemplu, în prezența declarației

var m : Matrice;

notația $M[1, 1]$ specifică componenta din linia 1, coloana 1 (*fig. 1.2*); notația $M[1, 2]$ specifică componenta din linia 1, coloana 2; notația $M[i, j]$ specifică componenta din linia i , coloana j .

Programul ce urmează afișează pe ecran suma componentelor variabilei **M** de tip **Matrice**. Valorile componentelor **M[1,1]**, **M[1,2]**, ..., **M[3,4]** se citesc de la tastatură.

```
Program P81;
{ Suma componentelor variabilei M de tip Matrice }
type Matrice = array [1..3, 1..4] of real;
var M : Matrice;
    i, j : integer;
    s : real;
begin
  writeln('Dati componentele M[i,j]:');
  for i:=1 to 3 do
    for j:=1 to 4 do
      begin
        write('M[', i, ', ', j, ']=');
        readln(M[i,j]);
      end;
    writeln('Ati introdus:');
    for i:=1 to 3 do
      begin
        for j:=1 to 4 do write(M[i,j]);
        writeln;
      end;
    S:=0;
    for i:=1 to 3 do
      for j:=1 to 4 do
        S:=S+M[i,j];
    writeln('Suma=', S);
    readln;
  end.
```

În general, un tip **tablou n-dimensional** ($n = 1, 2, 3$ etc.) se definește cu ajutorul diagramelor sintactice din *figura 1.3*. Atributul **packed** (împachetat) indică cerința de optimizare a spațiului de memorie pentru elementele tipului **array**. Menționăm că în majoritatea compilatoarelor actuale utilizarea acestui atribut nu are niciun efect, întrucât optimizarea se efectuează în mod automat.

Fiind date două variabile de tip tablou de același tip, numele variabilelor pot apărea într-o instrucțiune de atribuire. Această atribuire înseamnă copierea tuturor componentelor din membrul drept în membrul stâng.

De exemplu, în prezența declarațiilor

```
var a, b : Matrice;
```

instrucțiunea

```
a:=b
```

este corectă.

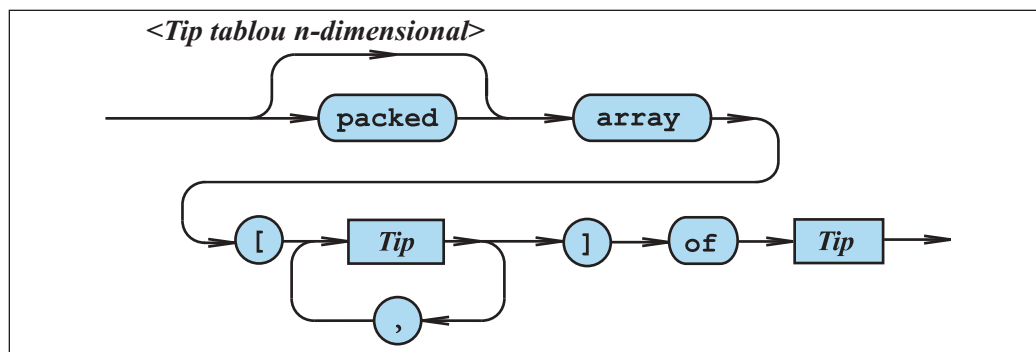


Fig. 1.3. Diagrama sintactică <Tip tablou n-dimensional>

În exemplele de mai sus tipul de bază (tipul componentelor) a fost de fiecare dată un tip simplu. Deoarece tipul de bază poate fi, în general, un tip arbitrar, devine posibilă definirea tablourilor cu componente de tip structurat. Considerăm acum un exemplu în care tipul de bază este el însuși un tip **array**:

```
Type Linie = array [1..4] of real;
      Tabel = array [1..3] of Linie;
var L : Linie;
    T : Tabel;
    x : real;
```

Variabila T este formată din 3 componente: T[1], T[2] și T[3] de tipul Linie. Variabila L este formată din 4 componente: L[1], L[2], L[3] și L[4] de tipul real. Prin urmare, atribuiriile

```
L[1] := x; x := L[3]; T[2] := L; L := T[1]
```

sunt corecte.

Elementele variabilei T pot fi specificate prin T[i][j] sau prescurtat T[i, j]. Aici i indică numărul componente de tip Linie în cadrul variabilei T, iar j – numărul componente de tip real în cadrul componente T[i] de tip Linie.

Subliniem faptul că declarațiile de forma

```
type array [T1, T2] of T3
```

și

```
type array [T1] of array [T2] of T3
```

definesc tipuri distincte de date. Prima declarație definește tablouri bidimensionale cu componente de tipul T₃. A doua declarație definește tablouri unidimensionale cu componente de tipul **array** [T₂] **of** T₃.

În programele PASCAL tablourile se utilizează pentru a grupa sub un singur nume mai multe variabile cu caracteristici identice.

C++

În limbajul de programare C++, un tip de date *tablou unidimensional* se definește printr-o construcție de forma:

typedef <Tip componente> <Nume tip tablou> [<Număr componente>];

În C++, un tip de date *tablou unidimensional* se definește printr-o construcție de forma

<Tip> <Nume_tablou> [<Nr_componente>];

unde <Tip componente> este tipul componentelor tabloului, care poate fi aproape orice tip de date, iar <Număr componente> indică numărul de componente ale tabloului.

Indicii componentelor tabloului pot lua doar valori consecutive întregi, începând cu zero: 0, 1, 2, ..., <Număr componente> -1.

Exemple:

- 1) **typedef int** Vector[5];
Vector V;
- 2) **typedef char** Simbol[10];
Simbol S;
- 3) **typedef float** Acceleratie[45];
Acceleratie A;

Structura datelor din exemplele în studiu este prezentată în *figura 1.1**.

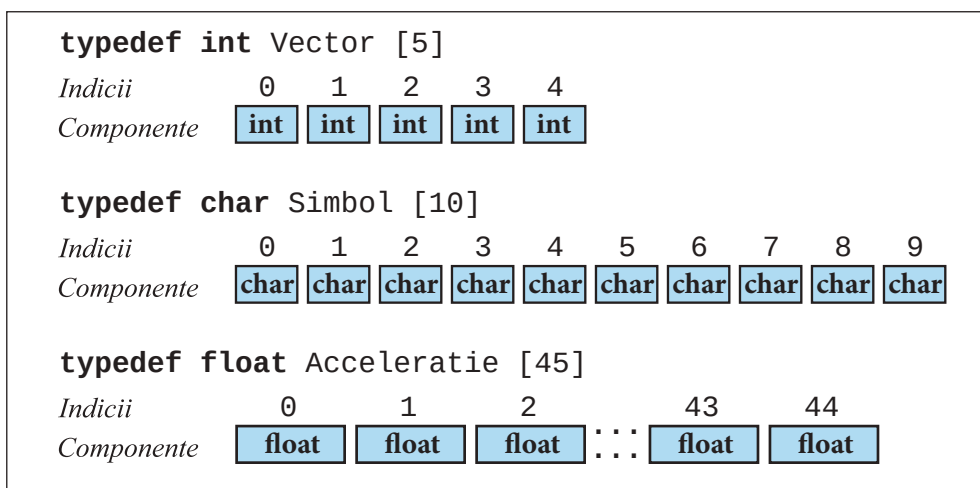


Fig. 1.1* Structura datelor de tip Vector, Simbol și Acceleratie

Fiecare componentă a unei variabile de tip *tablou unidimensional* poate fi specificată explicit, prin numele variabilei urmat de indicele respectiv încadrat de paranteze pătrate. Indicii trebuie să fie expresii întregi.

Exemple:

- 1) V[1], V[4];
- 2) S[0], S[5], S[9];
- 3) A[0], A[44], A[23];

Asupra componentelor datelor de tip *tablou unidimensional* se pot efectua toate operațiile admise de tipul acestora. Programul ce urmează afișează pe ecran suma componentelor variabilei *V* de tip tablou unidimensional. Valorile componentelor *V[0]*, *V[1]*, ..., *V[4]* se citesc de la tastatură.

```
// Program P78
// Suma componentelor variabilei V de tip Vector
// Numar fix de componente
#include <iostream>
using namespace std;
int main()
{
    typedef int Vector[5];
    Vector V;
    int i, S;
    cout << "Dati 5 numere intregi:" << endl;
    for (i=0; i<5; i++) cin>>V[i];
    cout << "Ati introdus:" << endl;
    for (i=0; i<5; i++) cout << V[i] << ' ';
    cout << endl;
    S=0;
    for (i=0; i<5; i++) S=S+V[i];
    cout << "Suma= " << S;
    return 0;
}
```

Pentru a extinde aria de aplicare a unui program, se recomandă ca numărul de componente ale datelor de tip *tablou* să fie specificate prin constante. Acest fapt oferă o flexibilitate mai mare programelor C++, ele putând fi aplicate pentru prelucrarea tablourilor, numărul de componente al cărora este necunoscut la momentul scrierii programului.

De exemplu, programul P78 poate fi modificat pentru a însuma n numere întregi, $n \leq 100$. Numărul concret de valori n ce vor fi stocate în tabloul unidimensional *V* se citește de la tastatură.

```
//Program P79
// Suma componentelor variabilei V de tip Vector
// Numar flotant de componente
#include <iostream>
using namespace std;
int main()
{
    const int nmax=100;
    typedef int Vector[nmax];
    Vector V;
    int n, i, S;
```

```

cout << "Dati n= "; cin >> n;
cout << "Dati "<<n<<" numere intregi:" << endl;
for (i=0; i<n; i++) cin >> V[i];
cout << "Ati introdus:" << endl;
for (i=0; i<n; i++) cout << V[i] << ' ';
cout << endl;
S=0;
for (i=0; i<n; i++) S=S+V[i];
cout << "Suma= " << S;
return 0;
}

```

În informatică, tablourile unidimensionale sunt utilizate foarte des pentru a sorta datele într-o anumită ordine, de exemplu, în cea crescătoare. Pentru o astfel de sortare se folosește următoarea metodă:

- 1) fiecare componentă a tabloului, începând cu prima, este comparată consecutiv cu fiecare dintre componentele ce urmează după ea;
- 2) dacă în urma comparării componentelor se constată că cea curentă este mai mare decât cea cu care este comparată, ele sunt schimbate cu locul;
- 3) procesul de parcurgere a tabloului continuă până când vor fi comparate ultima și penultima dintre componente.

În limbajul cotidian al informaticienilor, o astfel de sortare se numește *sortare prin metoda bulelor*, întrucât componentele tabloului își schimbă pozițiile ca bulele dintr-un lichid: cele mai ușoare se ridică la suprafață, iar cele mai grele se duc la fund.

Pentru exemplificare, în programul ce urmează, metoda bulelor este folosită pentru sortarea în ordine crescătoare a componentelor tabloului A. Componentele acestui tablou sunt citite de la tastatură, iar sortarea propriu-zisă se face în tabloul B.

```

// Program P80
// Sortarea prin metoda bulelor
#include <iostream>
using namespace std;
int main()
{
    const int nmax= 100;
    typedef int Tablou[nmax];
    Tablou A, B;
    int n, i, j;
    int x;
    cout<<"Dati numarul de componente n= ";
    cin>>n;
    cout<<"Dati componentele tabloului A: \n";
    for (i=0; i<n; i++) cin>>A[i];

```

```

for (i=0; i<n; i++) B[i]=A[i];
for (i=0; i<n-1; i++)
    for (j=i+1; j<n; j++)
        if (B[i]>B[j])
            { x=B[i]; B[i]=B[j]; B[j]=x;}
cout<<"Tabloul initial: \n";
for (i=0; i<n; i++) cout<<A[i]<<" ";
cout << endl;
cout<<"Tabloul sortat: \n";
for (i=0; i<n; i++) cout<<B[i]<<" ";
cout << endl;
return 0;
}

```

Un tip de date *tablou bidimensional* se definește cu ajutorul construcției gramaticale:

typedef <Tip componente> <Nume tip tablou>[<Număr linii>][<Număr coloane>];

Pentru exemplificare, în *figura 1.2** este prezentată structura datelor tipului Matrice:

```
typedef double Matrice[3][4];
```

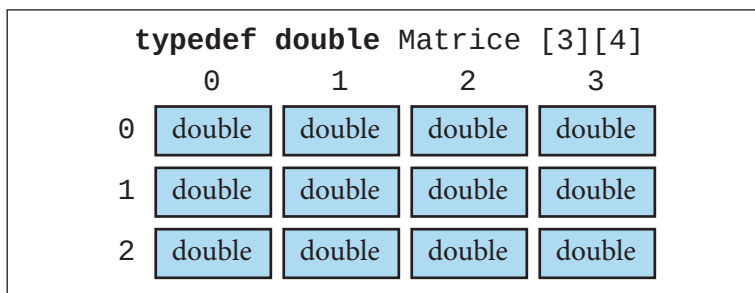


Fig. 1.2 *. Structura datelor de tip Matrice

Componentele unei variabile de tip *tablou bidimensional* se specifică explicit prin numele variabilei urmat de indicii respectivi încadrați de paranteze pătrate.

```
Matrice M;
```

notația $M[1,1]$ specifică componenta din linia 1, coloana 1 (*fig. 1.2**); notația $M[1,2]$ specifică componenta din linia 1, coloana 2; notația $M[i,j]$ specifică componenta din linia i , coloana j .

Programul ce urmează afișează pe ecran suma componentelor variabilei M de tip **Matrice**. Valorile componentelor $M[0,0]$, $M[0,1]$, ..., $M[2,3]$ se citesc de la tastatură.

```

// Program P81
// Suma componentelor variabilei M de tip Matrice
#include<iostream>
using namespace std;
int main()
{
    typedef double Matrice[3][4];
    Matrice M;
    int i, j;
    double S;
    cout << "Dati componentele M[i,j]: " << endl;
    for (i=0; i<3; i++)
        for (j=0; j<4; j++)
        {
            cout << "M[" <<i<< ', ' << j << "]= ";
            cin >> M[i][j];
        }
    cout << "Ati introdus:" << endl;
    for (i=0; i<3; i++)
    {
        for (j=0; j<4; j++) cout << M[i][j] << ' ';
        cout << endl;
    }
    S=0;
    for (i=0; i<3; i++)
        for (j=0; j<4; j++)
            S=S+M[i][j];
    cout << "Suma= " << S << endl;
    return 0;
}

```

În momentul declarării, elementele tabloului pot fi inițializate, de exemplu:

```
int T[4]={2,5,9,-4};
```

```
int M[3][2]={{9,7}, {-12,8}, {0,-4}};
```

Accentuăm faptul că în limbajul C++ indicii componentelor tabloului pot lua doar valori consecutive întregi, începând cu zero: 0, 1, 2, ..., <Număr componente> -1. Uneori, acest lucru poate crea anumite incomodități.

Astfel, în cazul exemplului referitor la consumul zilnic de energie electrică pe durata unei luni (vezi paragraful precedent), tabelul respectiv poate fi reprezentat pe calculator cu ajutorul următorului tip de date *tablou unidimensional*:

```
typedef float Consum[31];
Consum C;
```

Indicele tabloului C poate lua doar valorile 0, 1, 2, ... 30, iar componentele tabloului se specifică prin C[0], C[1], ..., C[30]. Evident, ultimei zile a lunii,

a 31-a, nu-i va corespunde nicio componentă a tabloului, iar tentativa de a utiliza specificarea `C[31]` ar putea declanșa, în anumite situații, o eroare de execuție. Indiscutabil, am putea să numerotăm zilele lunii începând nu cu „1”, ci cu „0”, însă acest lucru poate face programele C++ mai puțin intuitive.

Pentru a evita astfel de situații, se recomandă majorarea numărului de componente ale unui tablou cu o unitate față de cel cerut de datele structurate ce trebuie reprezentate pe calculator.

De exemplu, în cazul consumului zilnic de energie electrică pe durata unei luni, putem utiliza următorul tip de date:

```
typedef float Consum[32];
Consum C;
```

în care primei zile a lunii îi va corespunde componenta `C[1]`, zilei a doua — componenta `C[2]`, zilei a treia — componenta `C[3]` ș.a.m.d., iar componenta `C[0]` va rămâne neutilizată.

Într-un mod similar, pentru reprezentarea pe calculator a consumului zilnic de energie electrică pe parcursul unui an, poate fi folosit următorul tip de date:

```
typedef float Consum[13][32];
Consum C;
```

în care zilelor lunii *Ianuarie* le vor corespunde componentele `C[1, 1], ..., C[1, 31]`; zilelor lunii *Februarie* — componentele `C[2, 1], ..., C[2, 31]`, zilelor lunii *Decembrie* — componentele `C[12, 1], ..., C[12, 31]`.

Evident, unele componente ale tabloului `C`, în special cele din rândul și coloana 0, vor rămâne neutilizate. Însă întrucât lizibilitatea și corectitudinea programelor sunt cu mult mai importante decât neutilizarea anumitor componente, majoritatea informaticienilor consideră că acest fapt nu înrăutățește calitatea programelor C++.

Întrebări și exerciții

- ❶ **OBSERVĂ!** Precizați tipul indicilor și tipul componentelor din următoarele declarații:

PASCAL

```
type
P = array [1..5] of integer;
Culoare = (Galben, Verde,
Albastru, Violet);
R = array [Culoare] of real;
S = array [Culoare, 1..3]
of boolean;
T = array [boolean]
of Culoare;
```

C++

```
typedef int P[5];
enum Culoare {Galben, Verde,
Albastru, Violet};
typedef Culoare R[3];
typedef bool S[4][3];
typedef double T[2][5];
```

Reprezentați structura datelor de tipul `P`, `R`, `S` și `T` pe un desen (vezi figurile respective de mai sus).

- ❷ (PASCAL) Indicați pe diagrama sintactică din *figura 1.3* drumurile care corespund declarațiilor din exercițiul 1.

③ (PASCAL) Scrieți formulele metalingvistice care corespund diagramei sintactice <Tip tablou> din figura 1.3.

④ ANALIZEAZĂ! Identificați care dintre următoarele declarații sunt corecte:

PASCAL

C++

a: array [1..30] of integer;	int a[30];
a: array [1..10] of byte;	char a[1..10];
a: array [1.10..10.00] of real;	float a[1.10..10.00];
a: array [1..50] of char;	char a[50];

⑤ EXERSEAZĂ! Se consideră declarațiile:

PASCAL

C++

type Vector = array [1..5] of real; var x, y : Vector;	typedef double Vector[5]; Vector x,y;
---	---

Scrieți expresia aritmetică a cărei valoare este:

- suma primelor trei componente ale variabilei x;
- suma tuturor componentelor variabilei y;
- produsul tuturor componentelor variabilei x;
- valoarea absolută a componentei a treia a variabilei y;
- suma primelor componente ale variabilelor x și y.

⑥ ANALIZEAZĂ! Pentru următoarele declarații și instrucțiuni:

Declarație	Instrucțiune	Rezultat
PASCAL		
1) var a: array [1..5] of byte;	for i:=1 to 5 do a[i]:=i-1;	a) 1 1 1 1 1 b) 0 0 0 0 0 c) 0 1 2 3 4 d) 1 2 3 4 0
2) var v: array [0..4] of integer;	for i:=1 to 5 do v[i]:=2*(i-1);	a) 2 4 6 8 1 b) 0 2 4 6 8 c) 0 1 2 3 4 d) 1 2 3 4 5

C++		
1) unsigned char a[4];	for (i=0; i<5; i++) a[i]=i;	a) 1 1 1 1 1 b) 0 0 0 0 0 c) 0 1 2 3 4 d) 1 2 3 4 0
2) int v[4];	for (i=0; i<5; i++) v[i]=2*i;	a) 2 4 6 8 10 b) 0 2 4 6 8 c) 0 1 2 3 4 d) 1 2 3 4 5

Selectați din lista din partea dreaptă variantele ce conțin rezultatele corecte ale execuției fiecăreia dintre aceste instrucțiuni.

⑦ REZOLVĂ! Se consideră declarațiile:

PASCAL

C++

```
type Zi = (L, Ma, Mi, J, Vi, S, D);
    Venit = array [Zi] of real;
var v : Venit;
```

```
typedef double Venit[7];
Venit v;
```

Componentele variabilei *v* reprezintă venitul zilnic al unei întreprinderi. Elaborați un program care:

- calculează venitul săptămânal al întreprinderii;
- calculează media venitului zilnic;
- indică ziua în care s-a obținut cel mai mare venit;
- indică ziua cu venitul cel mai mic.

⑧ REZOLVĂ! Se consideră declarațiile:

PASCAL

C++

```
type Ora = 0..23;
    Grade = -40..40;
    Temperatura = array [Ora]
of Grade;
var t : Temperatura;
```

```
typedef int Ora
typedef int Temperatura [24];
Temperatura t;
```

Componentele variabilei *t* reprezintă temperaturile măsurate din oră în oră pe parcursul a 24 de ore. Elaborați un program care:

- calculează temperatura medie;
- indică maximul și minimul temperaturii;
- indică ora (orele) la care s-a înregistrat temperatura maximă;
- indică ora (orele) la care s-a înregistrat temperatura minimă;
- calculează numărul de zile în care au fost înregistrate temperaturi mai jos de zero grade;
- calculează numărul de zile în care au fost înregistrate temperaturi mai mari de media săptămânală.

⑨ Se consideră declarațiile:

PASCAL

C++

```
type Oras = (Chisinau, Orhei,
Balti, Tighina, Tiraspol);
    Zi=(L, Ma, Mi, J, Vi, S, D);
    Consum = array [Oras,Zi] of real;
var C : Consum;
    r : Oras;
    z : Zi;
```

```
enum Oras {Chisinau, Orhei,
    Balti, Tighina, Tiraspol};
enum Zi {L, Ma, Mi, J, Vi, S, D};
typedef double Consum[5][7];
Consum C;
Oras r;
Zi z;
```

Componenta $C[r, z] / C[r][z]$ în C++ a variabilei *C* reprezintă consumul de energie electrică a orașului *r* în ziua *z*. Elaborați un program care:

- calculează energia electrică consumată de fiecare oraș pe parcursul unei săptămâni;
- calculează energia electrică consumată zilnic de orașele în studiu;
- indică orașul cu un consum săptămânal maxim;
- indică orașul cu un consum săptămânal minim;
- indică ziua în care orașele consumă cea mai multă energie electrică;
- indică ziua în care orașele consumă cea mai puțină energie electrică.

⑩ Se consideră declarațiile

PASCAL

```
type Vector = array [1..5] of real;
Matrice = array [1..3, 1..4] of
boolean;
Linie = array [1..4] of real;
Tabel = array [1..3] of Linie;
var V : Vector;
    M : Matrice;
    L : Linie;
    T : Tabel;
    x : real;
    i : integer;
```

C++

```
typedef float Vector[5];
typedef bool Matrice[3][4];
typedef float Linie[4];
typedef Linie Tabel[3];
Vector V;
Matrice M;
Linie L;
Tabel T;
float x;
int i;
```

Care dintre atribuirile ce urmează sunt corecte?

PASCAL		C++	
a)	T[3]:=T[1]	a)	T[3]=T[1]
b)	M:=T	b)	M=T
c)	L:=V	c)	L=V
d)	L[3]:=x	d)	L[3]=x
e)	x:=i	e)	x=i
f)	i:=x	f)	i=x
g)	L[3]:=i	g)	L[3]=i
h)	i:=M[1,2]	h)	i=M[1][2]
i)	x:=V[4]	i)	x=V[4]
j)	L[3]:=V[4]	j)	L[3]=V[4]
k)	T[1]:=4	k)	T[1]=4
l)	T[2]:=V	l)	T[2]=V
m)	L:=T[3]	m)	L=T[3]
n)	T[1,2]:=M[1,2]	n)	T[1][2]=M[1][2]
o)	T[2,1]:=M[1,2]	o)	T[2][1]=M[1][2]
p)	M[1]:=4	p)	M[1]=4
q)	M[1,3]:=L[2]	q)	M[1][3]=L[2]
r)	x:=T[1][2]	r)	x=T[1,2]
s)	x:=M[1]	s)	x=M[1]
t)	L:=M[1]	t)	L=M[1]
u)	V[5]:=M[3,4]	u)	V[5]=M[3][4]
v)	L:=M[3,4]	v)	L=M[3][4]

- ⑪ **REZOLVĂ!** De la tastatură se citesc elementele unui tablou unidimensional, format din n numere întregi, $n < 10$. Elaborați un program care:
- afișează pe ecran componentele tabloului la un interval de 5 poziții;
 - afișează pe ecran numerele în ordinea inversă a introducerii în calculator;
 - sortează componentele tabloului în ordine descrescătoare;
 - afișează pe ecran doar componentele pare;
 - afișează pe ecran media aritmetică a componentelor pare;
 - afișează pe ecran doar componentele impare;
 - afișează pe ecran doar componentele care sunt mai mari ca x și nu sunt divizibile cu y (valorile x și y se citesc de la tastatură);
 - afișează pe ecran doar componentele care sunt mai mari ca x și mai mici decât y (valorile x și y se citesc de la tastatură);
 - afișează pe ecran pozițiile (indicii) componentelor impare negative;
 - afișează pe ecran pozițiile (indicii) componentelor ce conțin doar două cifre semnificative;
 - înlocuiește prima componentă a tabloului cu componenta de valoare minimă din tabloul respectiv;
 - înlocuiește componenta de valoare minimă din tabloul respectiv cu prima componentă a acestuia;
 - crează un tablou nou, format doar din componentele pare ale tabloului introdus de la tastatură;
 - crează un tablou nou, format doar din componentele divizibile cu 3 ale tabloului introdus de la tastatură;
 - crează un tablou nou, format doar din acele componente ale tabloului introdus de la tastatură care au cel mult patru divizori.
- ⑫ **REZOLVĂ!** O persoană intenționează să cumpere n produse distincte, numerotate prin $i = 1, 2, 3, \dots, n$. Fiecare dintre aceste produse poate fi procurat în oricare dintre cele k magazine disponibile, notate prin $j = 1, 2, 3, \dots, k$. Evident, cumpărătorul optează pentru magazinele cu cele mai mici prețuri. Elaborați un program care determină pentru fiecare produs i magazinul j în care el trebuie cumpărat și costul total C al cumpărăturilor făcute în acest mod.
Date de intrare: tabloul bidimensional P , în care componenta $P[i][j]$ reprezintă prețul produsului i în magazinul j .
Date de ieșire: costul total al cumpărăturilor C și tabloul unidimensional M , în care componenta $M[i]$ reprezintă magazinul j în care va fi procurat produsul i .
Restricții: $1 \leq n \leq 10$. $1 \leq k \leq 5$. Prețurile sunt indicate prin numere reale, cu două cifre după punctul zecimal.
- ⑬ **STUDIU DE CAZ.** Proiectați structurile de date de tip tablou necesare pentru reprezentarea în programele PASCAL/C++ a tabelelor ce conțin informațiile referitoare la consumul zilnic de energie electrică, respectiv, pe durata unei luni și pe durata unui an, descrise în paragraful precedent. Analizați cum numărul de componente din declarațiile de tablouri influențează înțelegerea programelor de către utilizatori.
- ⑭ **ÎNVAȚĂ SĂ ÎNVEȚI!** Tablourile bidimensionale cu același număr de linii și coloane se numesc *tablouri pătratice*. Mulțimea componentelor care au indicele de linie egal cu indicele de coloană formează *diagonala principală* a unui tablou pătratic. Evident, această mulțime poate fi simbolizată printr-o linie imaginară ce reunește componenta din colțul stânga-sus cu componenta din colțul dreapta-jos al tabloului pătratic. Într-un mod similar, poate fi definită și diagonala secundară a tabloului pătratic, care, de asemenea, poate fi simbolizată printr-o

linie imaginară ce reunește componenta din colțul stânga-jos cu componenta din colțul dreapta-sus.

Elaborați un program care citește de la tastatură un tablou pătratic cu n linii, $2 \leq n \leq 10$, și afișează la ecran suma componentelor care se află:

- a) pe diagonala principală;
- b) pe diagonala secundară;
- c) mai sus de diagonala principală;
- d) mai jos de diagonala principală;
- e) mai sus de diagonala secundară;
- f) mai jos de diagonala secundară.

Se consideră că componentele tabloului pătratic sunt numere întregi, care se citesc de la tastatură.

- 15 DESCOPERĂ! Limbajul C++ oferă un tip alternativ de tablouri bidimensionale, definit în biblioteca `<array>`. Explorați această bibliotecă și rescrieți programele din acest paragraf folosind oportunitățile oferite de ea. Prin ce se aseamănă și prin ce diferă aceste două moduri de definire a tablourilor?

1.3. Tipuri de date șir de caractere

Metodele de reprezentare și de prelucrare a șirurilor de caractere au o importanță deosebită, întrucât foarte multe obiecte din lumea reală sunt descrise nu doar prin numere, ci și prin cele mai diverse texte. De obicei, în mediile moderne de dezvoltare a programelor, șirurile de caractere pot fi reprezentate prin mai multe structuri de date. În continuare vom studia următoarele tipuri de date utilizate pentru reprezentarea șirurilor de caractere:

- tablourile unidimensionale cu componente de tipul `char` (caractere);
- *string*-urile.

Amintim că în cazul limbajelor de programare, cuvântul englez *string* are semnificația de șir, serie, succesiune de caractere.

PASCAL

Șiruri de caractere de tipul *tablouri unidimensionale*

În limbajul-standard tipul de date *șir de caractere* reprezintă un caz special al tipului **array** și se definește printr-o construcție de forma

`<Nume tip> ::= packed array [1..n] of char;`

Mulțimea de valori ale tipului de date în studiu este formată din toate șirurile ce conțin exact n caractere.

Exemplu:

```
Program P82;
{ Șiruri de caractere de tip tablou unidimensional }
type      Nume = packed array [1..8] of char;
           Prenume = packed array [1..5] of char;
var      N : Nume;
           P : Prenume;
```

```

begin
  N:='Munteanu';
  P:='Mihai';
  writeln(N);
  writeln(P);
  readln;
end.

```

Rezultatul afișat pe ecran:

```

Munteanu
Mihai

```

Întrucât șirurile de lungime diferită aparțin unor tipuri distincte de date, în cadrul programului P82 nu sunt admise atribuiri de genul:

```

N:= 'Olaru';
P:= 'Ion'.

```

În astfel de cazuri, programatorul va completa datele respective cu spațiu până la numărul stabilit de caractere *n*, de exemplu:

```

N:= 'Olaru ';
P:= 'Ion '.

```

Valorile unei variabile *v* de tip **packed array** [1..*n*] of char pot fi introduse de la tastatură numai prin citirea separată a componentelor respective:

```

read(v[1]); read(v[2]); ...; read(v[n]).

```

În schimb, o astfel de valoare poate fi afișată în totalitatea ei printr-un singur apel `write(v)` sau `writeln(v)`.

Accentuăm faptul că șirurile de caractere de tip **packed array** [1..*n*] of char conțin exact *n* caractere, adică sunt **șiruri de lungime constantă**. Evident, lungimea lor nu poate fi modificată pe parcursul derulării programului respectiv. Acest fapt complică elaborarea programelor destinate prelucrării unor șiruri arbitrare de caractere.

Șiruri de caractere de tipul **string**

Pentru a ușura munca programatorilor, versiunile actuale ale limbajului PASCAL permit utilizarea tipului de date **string**, mulțimea de valori a căruia este formată din **șiruri de caractere de lungime variabilă**. Un astfel de tip se declară printr-o construcție de forma:

```

type <Nume tip> = string; sau type <Nume tip> = string [nmax];

```

unde *nmax* este lungimea maximă pe care o pot avea șirurile respective. În lipsa parametrului *nmax* lungimea maximă se stabilește implicit, în mod obișnuit – 255 de caractere.

Asupra șirurilor de tip **string** se poate efectua operația de concatenare (juxtapunere), notată prin semnul „+”. Lungimea curentă a unei valori *v* de tip **string** poate fi aflată cu ajutorul funcției predefinite `length(v)` care returnează

o valoare de tip **integer**. Indiferent de lungime, toate șirurile de caractere de tip **string** sunt compatibile.

Exemplu:

```
Program P83_a;
{ Siruri de caractere de tip string }
type Nume = string [8];
      Prenume = string [5];
      NumePrenume = string;
var N : Nume;
    P : Prenume;
    NP : NumePrenume;
    L : integer;
begin
  N:='Munteanu';    L:=length(N);    writeln(N, L:4);
  P:='Mihai';        L:=length(P);    writeln(P, L:4);
  NP:=N+' '+P;        L:=length(NP);  writeln(NP, L:4);
  N:='Olaru';         L:=length(N);    writeln(N, L:4);
  P:='Ion';           L:=length(P);    writeln(P, L:4);
  NP:=N+' '+P;        L:=length(NP);  writeln(NP, L:4);
  readln;
end.
```

Rezultatele afișate pe ecran:

```
Munteanu   8
Mihai      5
Munteanu Mihai  14
Olaru      5
Ion        3
Olaru Ion   9
```

Se observă că pe parcursul derulării programului în studiu lungimea șirurilor de caractere N, P și NP se schimbă.

Asupra șirurilor de caractere sunt admise operațiile relaționale <, <=, =, >=, >, <>. Șirurile se compară componentă cu componentă, de la stânga la dreapta, în conformitate cu ordonarea caracterelor în tipul de date **char**. Ambii operanzi trebuie să fie de tip **packed array [1..n] of char** cu același număr de componente sau de tip **string**. Evident, operanzii de tip **string** pot avea lungimi arbitrare.

De exemplu, rezultatul operației

```
'AC' < 'BA'
```

este **true**, iar rezultatul operației

```
'AAAAC' < 'AAAAB'
```

este **false**.

O variabilă de tip *șir de caractere* poate fi folosită fie în totalitatea ei, fie parțial, prin referirea unui caracter din șir.

De exemplu, pentru $P = \text{'Mihai'}$ avem $P[1] = \text{'M'}$, $P[2] = \text{'i'}$, $P[3] = \text{'h'}$ ș.a.m.d. După executarea secvenței de instrucțiuni

```
P[1] := 'P';  
P[2] := 'e';  
P[3] := 't';  
P[4] := 'r';  
P[5] := 'u'
```

variabila P va avea valoarea 'Petru' .

Programul ce urmează citește de la tastatură șiruri arbitrare de caractere și afișează pe ecran numărul de spații în șirul respectiv. Derularea programului se termină după introducerea șirului 'Sfarsit' .

```
Program P83_b;  
{ Numarul de spatii într-un sir de caractere }  
var S : string;  
    i, j : integer;  
begin  
    writeln('Dati siruri de caractere:');  
    repeat  
        readln(S);  
        i:=0;  
        for j:=1 to length(S) do  
            if S[j]=' ' then i:=i+1;  
        writeln('Numarul de spatii=', i);  
    until S='Sfarsit';  
end.
```

Menționăm faptul că compilatoarele moderne ale limbajului PASCAL conțin mai multe funcții standard de prelucrare a șirurilor de caractere, de exemplu, concatenarea, copierea, secvențierea, inserarea sau ștergerea anumitor caractere din șir ș.a.m.d. Descrierea acestor funcții poate fi găsită în sistemele de asistență ale mediilor integrate de dezvoltare a programelor.

C++

Șiruri de caractere de tipul *tablouri unidimensionale*

În programele C++ șirurile de caractere pot fi reprezentate prin *tablouri unidimensionale*, componentele cărora sunt de tip **char**. Astfel de tipuri se declară cu ajutorul construcției gramaticale de forma:

```
typedef char <Nume tip șir> [nmax];
```

unde $nmax$ indică numărul maxim de caractere pe care îl pot avea șirurile de tipul respectiv.

Mulțimea de valori ale tipului de date $\langle \text{Nume tip șir} \rangle$ este formată din șiruri ce conțin până la $nmax$ caractere. Întrucât în reprezentarea internă una dintre componentele tabloului unidimensional ce reprezintă șirul de caractere se utilizează pentru a indica sfârșitul șirului, lungimea șirului va fi de cel mult $nmax-1$ caractere semnificative.

Exemple:

- 1) `typedef char Nume[20];`
`Nume N;`
- 2) `typedef char Prenume[10];`
`Prenume P;`
- 3) `typedef char Companie[45];`
`Companie C;`

După cum a fost menționat mai sus, în reprezentarea internă, sfârșitul de șir se indică cu ajutorul unui simbol special, și anume, simbolul *escape* „\0”.

De exemplu, dacă variabila P de tip Prenume conține șirul de caractere „Ion”, tabloul respectiv are forma:

0	1	2	3	4	5	6	7	8	9
I	o	n	\0						

Dacă însă aceeași variabilă P va conține șirul de caractere "Cristina", tabloul respectiv va avea forma:

0	1	2	3	4	5	6	7	8	9
C	r	i	s	t	i	n	a	\0	

Prin urmare, în pofida faptului că în declarația tipului de date Prenume sunt indicate 10 caractere, șirurile de acest tip pot conține până la 9 caractere semnificative.

În principiu, șirurile de caractere reprezentate prin tablouri unidimensionale pot fi prelucrate accesând componentele dorite ale acestora.

De exemplu, dacă se dorește înscrierea în variabila P a șirului de caractere "Ion", în acest scop pot fi folosite următoarele instrucțiuni:

```
P[0] = 'I';  
P[1] = 'o';  
P[2] = 'n';  
P[3] = '\0';
```

Accentuăm faptul că indicarea sfârșitului de șir cu ajutorul simbolului *escape* „\0” este obligatorie. În caz contrar, vor apărea erori de execuție.

Valorile variabilelor de tip șiruri de caractere – tablouri unidimensionale, pot fi afișate pe ecran cu ajutorul operatorului de transfer al datelor << în fluxul de ieșire cout.

Exemplu:

```
// Program P82_a  
// Șiruri de caractere de tip tablou unidimensional  
#include <iostream>  
using namespace std;  
int main()  
{
```

```

typedef char Nume[20];
Nume N;
typedef char Prenume[10];
Prenume P;
N[0]='L'; N[1]='u'; N[2]='p'; N[3]='u'; N[4] = '\0';
P[0]='I'; P[1]='o'; P[2]='n'; P[3]='\0';
cout << N << endl;
cout << P << endl;
return 0;
}

```

Rezultatele afișate pe ecran:

```

Lupu
Ion

```

La citirea șirurilor de caractere de la tastatură se va ține cont de faptul că intrarea standard este privită ca o sursă ce furnizează fluxul încontinuu de caractere cîn. Pentru a putea extrage din acest flux șirurile dorite, programul trebuie să cunoască care caracter le delimitează. De obicei, în calitate de *delimitator* se folosesc așa-numitele *caractere albe* (spațiul, tabulatorul ↵, sfârșitul de linie), însă programatorul poate utiliza și alte caractere.

Pentru citirea șirurilor de caractere **delimitate prin caractere albe** se folosește operatorul >>. Pentru afișarea șirurilor de caractere se folosește operatorul <<.

Exemplu:

```

// Program P82_b
// Citirea sirurilor de caractere
// Delimitatori: spatiu sau sfarsit de linie
#include <iostream>
using namespace std;
int main()
{
    typedef char Nume[20];
    Nume N;
    typedef char Prenume[10];
    Prenume P;
    cin >> N; // citirea sirului N
    cin >> P; // citirea sirului P
    cout << N << endl; // afisarea sirului N
    cout << P << endl; // afisarea sirului P
    return 0;
}

```

Presupunem că după lansarea programului în execuție, utilizatorul tastează:

```

Margine<ENTER>
Cristina<ENTER>

```

În acest caz, fluxul de intrare `cin` va avea forma:

```
Margine<ENTER>Cristina<ENTER>
```

↑

unde simbolul ↑ reprezintă cursorul acestuia. Inițial, cursorul se află la începutul fluxului, adică în fața caracterului `M`.

În procesul executării instrucțiunii `cin>>N`, programul, începând cu poziția curentă a cursorului, parcurge fluxul de intrare de la stânga la dreapta, mutând cursorul caracter după caracter și depunând caracterele citite în variabila `N`. Imediat ce cursorul ajunge la delimitatorul sfârșit de linie `<ENTER>`, instrucțiunea `cin>>N` înscrie în variabila `N` caracterul *escape* `'\0'` și termină citirea primului șir de caractere. Evident, după executarea instrucțiunii `cin>>N`, variabila `N` va primi valoarea "Margine", iar cursorul se va afla în fața caracterului `C`.

Într-un mod similar, instrucțiunea `cin>>P` continuă parcurgerea fluxului de intrare, mutând cursorul și depunând caracterele citite, unul după altul, în variabila `P`. Imediat ce cursorul ajunge la delimitatorul sfârșit de linie `<ENTER>`, instrucțiunea `cin>>P` înscrie în variabila `P` caracterul *escape* `'\0'` și termină citirea celui de al doilea șir de caractere. Evident, după executarea instrucțiunii `cin>>P`, variabila `P` va primi valoarea "Cristina", iar cursorul se va afla la sfârșitul fluxului.

Prin urmare, programul va afișa pe ecran:

```
Margine
Cristina
```

Accentuăm faptul că instrucțiunile de citire a datelor din fluxul de intrare `cin >>` folosesc în calitate de delimitator nu doar sfârșitul de linie `<ENTER>`, dar și spațiul. Astfel, dacă utilizatorul va tasta:

```
Margine Cristina Lupu Ion<ENTER>
```

↑

variabila `N` va primi valoarea "Margine", variabila `P` – valoarea "Cristina", iar cursorul se va afla în fața caracterului `L`. Restul caracterelor din fluxul de date, și anume "Lupu Ion", vor rămâne stocate în fluxul de intrare.

Pentru a citi șirurile de caractere **delimitate doar de sfârșit de linie**, adică de acționarea tastei `<ENTER>`, se folosește funcția `cin.getline`. Apelul acestei funcții are forma:

```
cin.getline (S, n);
```

unde:

`S` – variabila în care va fi depus șirul de caractere extras din fluxul de intrare;
`n` – numărul maxim de caractere ce pot fi citite din fluxul de intrare.

Exemplu:

```
// Program P82_c
// Citirea sirurilor de caractere
// Delimitator: sfarsitul de linie
#include <iostream>
```

```
using namespace std;
int main()
{
    typedef char NumePrenume[30];
    NumePrenume NP;
    cin.getline(NP, 30); // citirea sirului NP
    cout << NP << endl;
    return 0;
}
```

Dacă, în procesul executării programului de mai sus, utilizatorul va tasta:

```
Paduraru Elena - Promovata<ENTER>
```

variabila NP va primi valoarea "Paduraru Elena - Promovata".

O altă modalitate de citire a unui șir care poate conține spații este folosirea funcției `get()`.

Întrucât prelucrarea șirurilor de caractere prin accesarea fiecăreia dintre componentele tablourilor respective este anevoioasă, limbajul C++ conține un șir de funcții predefinite care simplifică acest proces. Prezentăm în continuare câteva dintre ele:

- `strcpy( $S_1, S_2$ )` – copie șirul de caractere S_2 în șirul S_1 ;
- `strcat( $S_1, S_2$ )` – alipește la șirul S_1 șirul de caractere S_2 ;
- `strlen( $S$ )` – returnează lungimea șirului de caractere S .

Mediile de dezvoltare ale programelor C++ conțin și alte funcții predefinite, destinate prelucrării șirurilor de caractere. Descrierea și modul de utilizare a acestor funcții pot fi găsite în sistemele de asistență ale mediilor respective.

Pentru a putea utiliza aceste funcții, în programul în curs de elaborare trebuie inclusă directiva

```
#include <cstring>;
```

Exemplu:

```
// Program P82_d
/* Functii predefinite pentru prelucrarea */
/* sirurilor de tip tablou unidimensional */
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    typedef char Nume[20];
    Nume N;
    typedef char Prenume[10];
    Prenume P;
    strcpy(N, "Munteanu");
    cout << N << "    " << strlen(N) << endl;
```

```

strcpy(P, "Mihai");
cout << P << " " << strlen(P) << endl;
strcat(N, " ");
cout << strcat(N, P) << " " << strlen(N) << endl;
return 0;
}

```

Rezultatele afișate pe ecran:

```

Munteanu   8
Mihai      5
Munteanu Mihai   14

```

Evident, în cazul reprezentării șirurilor de caractere prin tablouri unidimensionale, ele pot fi inițializate în momentul declarării. Se va ține însă cont de faptul că tablourile respective trebuie să conțină în mod obligatoriu caracterul *escape* '`\0`' (sfârșit de șir).

Exemple:

- 1) `char S[]={'I', 'o', 'n', '\0'};`
- 2) `char S[]="Ion";`
- 3) `char T[]={'E', 'l', 'e', 'n', 'a', '\0'};`

Șiruri de caractere de tipul `string`

Cunoaștem deja că șirurile de caractere de tip `char <Nume tip șir> [nmax]` pot conține cel mult `nmax-1` caractere semnificative. Însă, foarte des, la momentul scrierii programelor C++, valoarea `nmax` este necunoscută, fapt ce complică procesul de programare.

Pentru a ușura munca programatorilor, limbajul de programare C++ conține tipul predefinit de date `string`, care permite declararea șirurilor de caractere fără a indica lungimea maximă posibilă a acestora. Pentru a utiliza acest tip, în partea declarativă a programului C++ trebuie inserată directiva:

```
#include <string>;
```

Șirurile de caractere de tip `string` pot apărea în instrucțiunile de atribuire și asupra lor poate fi efectuată operația de concatenare (de alipire), notată prin simbolul „+”.

Exemplu:

```

// Program P83_a
// Șiruri de caractere de tip string
#include <iostream>
#include <string>
using namespace std;
int main()
{

```

```

string N;           // Nume
string P;           // Prenume
string NP;          // Nume si Prenume
N = "Olaru";        // atribuire
P = "Andrei";       // atribuire
NP = N + " " + P;   // concatenare si atribuire
cout << N << endl;
cout << P << endl;
cout << NP << endl;
return 0;
}

```

Acest program va afișa pe ecran:

```

Olaru
Andrei
Olaru Andrei

```

Asupra șirurilor de caractere sunt admise următoarele operații relaționale: <, <=, ==, >=, >, !=. Șirurile se compară componentă cu componentă, de la stânga la dreapta, în conformitate cu ordonarea caracterelor în tipul de date `char`.

De exemplu, rezultatul operației

```
'AC' < 'BA'
```

este `true`, iar rezultatul operației

```
'AAAAC' < 'AAAAB'
```

este `false`.

O variabilă de tip șir de caractere poate fi folosită fie în totalitatea ei, fie parțial, prin referirea unui caracter din șir.

De exemplu, după executarea secvenței de instrucțiuni

```

P="Mihai";
P[0]='P'; P[1]='e'; P[2]='t'; P[3]='r'; P[4]='u';

```

variabila `P` va avea valoarea "Petru".

De obicei, citirea unei linii introduse de la tastatură într-o variabilă `S` de tip `string` se efectuează cu ajutorul unui apel de forma:

```
getline(cin, S);
```

Lungimea `l` a unui șir de caractere `S` de tip `string` poate fi aflată cu ajutorul unei instrucțiuni de forma:

```
l = S.length();
```

unde `S` este șirul de caractere supus prelucrării.

Programul ce urmează citește de la tastatură șiruri arbitrare de caractere și afișează pe ecran numărul de spații în fiecare șir citit. Derularea programului se termină după introducerea șirului 'Sfarsit'.

```

// Program P83_b
// Numarul de spatii intr-un sir
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string Sir; // Sirul citit de la tastatura
    int L;      // Lungimea sirului citit
    int n;      // Numarul de spatii in sirul citit
    int i;

    Repeta:
    cout << "Dati sirul de caractere:" << endl;
    getline(cin, Sir); // Citirea sirului
    L = Sir.length();  // Lungimea sirului
    cout << "Lungimea sirului = " << L << endl;
    n = 0;
    for (i=0; i < L; i++)
        if (Sir[i] == ' ') n++;
    cout << "Numarul de spatii in sir = " << n << endl;
    cout << endl;
    if (Sir != "Sfarsit") goto Repeta;
    return 0;
}

```

Bibliotecile standard ale limbajului C++ oferă programatorului posibilitatea să utilizeze ambele reprezentări ale șirurilor de caractere, atât prin tablouri, cât și prin *string*-uri.

Transformarea din *string* în tablou se poate face cu ajutorul unui apel de forma:

`S.c_str();`

unde *S* este o variabilă de tip *string*.

Transformarea din tablou de caractere în *string* se poate face printr-o simplă atribuire de forma *S = T*.

Exemplu:

```

// Program P83_c
// Transformari string <--> tablou de caractere
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char A[20]; // A este un tablou de caractere
    string B;   // B este un string
    // Transformarea string-ului B in tabloul A
    B="Informatica";
}

```

```

strcpy(A, B.c_str());
cout << A << endl;
// Transformarea tabloului A in string-ul B
A[0]='A'; A[1]='d'; A[2]='m'; A[3]='i'; A[4]='s'; A[5]='\0';
B=A;
cout << B << endl;
return 0;
}

```

Întrebări și exerciții

- 1 Cum se definește un tip de date *șir de caractere*? Care sunt cele două modalități de reprezentare a șirurilor de caractere în programele PASCAL și C++?
- 2 Ce operații pot fi efectuate asupra șirurilor de caractere reprezentate prin tablouri unidimensionale? Asupra șirurilor de caractere de tipul *string*?
- 3 **PROGRAMEAZĂ!** Elaborați un program care propune elevului să introducă de la tastatură, în două linii:
 - numele și prenumele elevului/eleveii;
 - profesia la care el/ea aspiră.
 Folosiți mesaje sugestive și organizați dialogul om–calculator conform următorului model:

```

Ce nume si prenume ai?
Paduraru Elena
Ce iti doresti sa devii in viitor?
medic
Te numesti Paduraru Elena
Doresti sa devii medic

```

În acest model datele introduse de utilizator sunt evidențiate prin îngroșare.

- 4 **OBSERVĂ!** Comentați următorul program:

PASCAL

```

Program P84;
{ Eroare }
var S : packed array [1..5] of
char;
begin
  S:='12345';
  writeln(S);
  S:='Sfat';
  writeln(S);
end.

```

C++

```

//Program P84
// Eroare
#include <iostream>
using namespace std;
int main()
{
  char S[]="12345";
  cout<<S<<endl;
  S="Sfat";
  cout<<S;
  return 0;
}

```

- ⑤ **EXPERIMENTEAZĂ!** Inserați înaintea instrucțiunii „return 0;” din programul P82_b secvența de instrucțiuni:

```
cin >> N;
cin >> P;
cout << N << endl;
cout << P << endl;
```

Lansați în execuție programul modificat și tastați:

Margine Cristina Lupu Ion<ENTER>

Explicați mesajele afișate pe ecran.

- ⑥ **ANALIZEAZĂ!** Se consideră șiruri de caractere de tip string. Precizați rezultatul operațiilor relaționale:

PASCAL		C++	
a)	'B' < 'A'	a)	'B' < 'A'
b)	'BB' > 'AA'	b)	"BB" > "AA"
c)	'BAAAA' < 'AAAAA'	c)	"BAAAA" < "AAAAA"
d)	'CCCCD' > 'CCCCA'	d)	"CCCCD" > "CCCCA"
e)	'A' = 'AA'	e)	"A" == "AA"
f)	'BB' < 'B'	f)	"BB" < "B B"
g)	'A' = 'a'	g)	"A" == "a"
h)	'Aa' > 'aA'	h)	"Aa" > "aA"
i)	'123' = '321'	i)	"123" == "321"
j)	'12345' > '12345'	j)	"12345" > "12345"

- ⑦ **REZOLVĂ!** Elaborați un program care citește de la tastatură șirul de caractere *S* și afișează pe ecran:
- numărul de apariții ale caracterului 'A' în șirul *S*;
 - șirul obținut prin substituirea caracterului 'A' prin caracterul '*';
 - șirul obținut prin radierea din șirul *S* a tuturor aparițiilor caracterului 'B';
 - numărul de apariții ale silabei MA în șirul *S*;
 - șirul obținut prin substituirea tuturor aparițiilor în șirul *S* a silabei MA prin silaba TA;
 - șirul obținut prin radierea din șirul *S* a tuturor aparițiilor silabei TO;
 - scrierea inversă a șirului *S*;
 - true dacă șirul *S* este palindrom și false în caz contrar;
 - șirul obținut prin transformarea tuturor literelor mici din componența șirului *S* în litere mari;
 - șirul obținut prin transformarea primei litere a fiecăruia dintre cuvintele din componența șirului *S* în literă mare;
 - șirul obținut prin sortarea în ordine alfabetică a caracterelor din șirul *S*.
- ⑧ **PROGRAMEAZĂ!** Se consideră șiruri de caractere formate din literele mari ale alfabetului latin și spațiu. Elaborați un program care afișează șirurile în studiu după următoarele reguli:

- fiecare literă de la 'A' până la 'Y' se înlocuiește prin următoarea literă din alfabet;
- fiecare literă 'Z' se înlocuiește prin litera 'A' ;
- fiecare spațiu se înlocuiește prin ' - ' .

- ⑨ **PROGRAMEAZĂ!** Elaborați un program care descifrează șirurile cifrate conform regulilor din exercițiul precedent.
- ⑩ **PROGRAMEAZĂ!** Elaborați un program care citește de la tastatură șirul S format din n caractere, $n \leq 30$, și afișează pe ecran toate șirurile formate prin ștergerea din S a ultimelor k caractere, $k = 0, 1, 2, \dots, n - 1$. De exemplu, pentru $S = \text{Test}$, programul va afișa:

```
Test
Tes
Te
T
```

- ⑪ **CREEAZĂ!** Elaborați un program care:
- citește de la tastatură m , $m \leq 100$, șiruri de caractere, formate din literele mici ale alfabetului latin;
 - sortează șirurile în cauză în ordine alfabetică;
 - afișează șirurile sortate pe ecran.
- ⑫ **PROGRAMEAZĂ!** Șirul S este compus din câteva propoziții. Propozițiile se termină cu punct, semnul de exclamare sau semnul întrebării. Elaborați un program care afișează pe ecran:
- numărul de propoziții din șirul de caractere introdus de la tastatură;
 - numărul de cuvinte din componența fiecărei propoziții.
- ⑬ **ÎNVAȚĂ SĂ ÎNVEȚI!** Este cunoscut faptul că compilatoarele moderne conțin mai multe funcții predefinite destinate prelucrării șirurilor de caractere, de exemplu: concatenarea, copierea, secvențierea, inserarea sau ștergerea anumitor caractere ș.a.m.d. Descrierea acestor funcții poate fi găsită în sistemele de asistență ale mediilor integrate de dezvoltare a programelor. Studiați, în mod individual sau în echipă, funcțiile predefinite de prelucrare a șirurilor de caractere și completați tabelul:

Funcția	Descriere	Exemple
...		
...		

Utilizând funcțiile respective, rescrieți programele din acest paragraf. Estimați, în ce măsură utilizarea acestor funcții simplifică procesul de elaborare a programelor PASCAL/C++.

- ⑭ **EXPLOREAZĂ!** În afară de funcția predefinită `c.in.getline`, pentru citirea șirurilor de caractere poate fi folosită și funcția `c.in.get`. Studiați, în mod individual sau în echipă, această funcție. Aflați care sunt deosebirile dintre funcțiile `c.in.getline` și `c.in.get`. Determinați în care cazuri este indicată folosirea funcției `c.in.getline` și în care cazuri – folosirea funcției `c.in.get`. Utilizând funcția `c.in.get`, elaborați un program care citește de la tastatură un șir de caractere și afișează pe ecran lungimea acestuia. Șirul poate să includă unul sau mai multe spații, iar sfârșitul lui este indicat de caracterul 'x'.

1.4. Tipuri de date *articol*

Articolul, cunoscut în informatică și sub denumirea de *înregistrare* sau *structură*, reprezintă o grupare de date, reunite sub un singur nume. Articolele sunt formate din componente, denumite **câmpuri**. Spre deosebire de componentele unui tablou, câmpurile pot fi de tipuri diferite. Fiecare câmp are un nume (identificator de câmp).

De exemplu, rezultatele examenului de bacalaureat pot fi afișate în forma:

<i>Nume câmp</i> →	Nume	Prenume	Nota medie
	Margine	Cristina	10,0
<i>Articol</i> →	Munteanu	Ion	8,7
	...	...	...
	Păduraru	Elena	9,5

În acest model de afișare a rezultatelor examenului de bacalaureat, fiecărui elev îi corespunde un articol (o înregistrare) ce conține trei câmpuri intitulate *Nume*, *Prenume* și *Nota medie*. Pe calculator, datele din câmpurile *Nume* și *Prenume* pot fi reprezentate prin șiruri de caractere, iar cele din câmpul *Nota medie* – prin numere reale.

PASCAL

În limbajul PASCAL, un tip de date *articol* se definește printr-o structură gramaticală de forma:

```
type <Nume tip> = record
    <Nume câmp 1> :  $T_1$ ;
    <Nume câmp 2> :  $T_2$ ;
    ...
    <Nume câmp n> :  $T_n$ ;
end;
```

unde $T_1, T_2, \dots, T_n$ specifică tipul câmpurilor respective. Tipul unui nume de câmp este arbitrar, astfel un câmp poate să fie, la rândul lui, tot de tip *articol*. Prin urmare, se pot defini tipuri imbricate.

Exemple:

```
1) type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
var E1, E2 : Elev;

2) type Punct = record
    x : real; { coordonata x }
    y : real; { coordonata y }
end;
var P1, P2 : Punct;
```

```

3) type Triunghi = record
        A : Punct; { vârful A }
        B : Punct; { vârful B }
        C : Punct; { vârful C }
    end;
var T1, T2, T3 : Triunghi;

```

Structura datelor din exemplele în studiu este prezentată în *figura 1.4*.

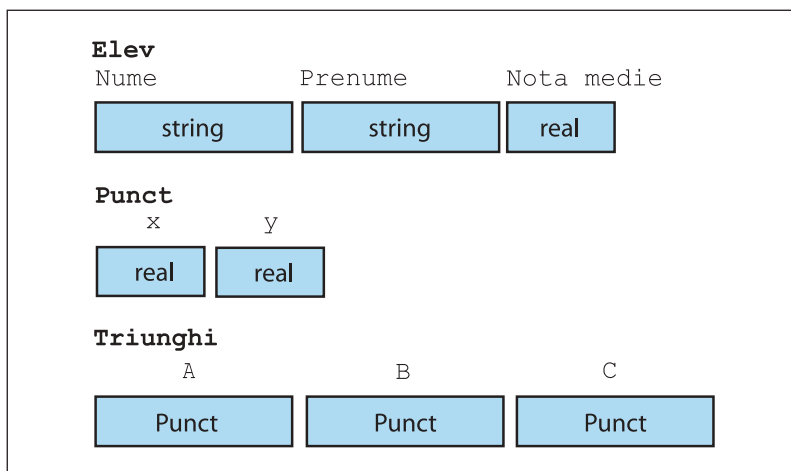


Fig. 1.4. Structura datelor de tip Elev, Punct și Triunghi

Fiind date două variabile de tip *articol* de același tip, numele variabilelor pot apărea într-o instrucțiune de atribuire. Această atribuire înseamnă copierea tuturor câmpurilor din membrul drept în membrul stâng. De exemplu, pentru tipurile de date și variabilele declarate mai sus instrucțiunile

```

E1:=E2;
T2:=T3;
P2:=P1

```

sunt corecte.

Fiecare componentă a unei variabile de tip **record** poate fi specificată explicit, prin numele variabilei și denumirile de câmpuri, separate prin puncte.

Exemple:

- 1) E1.Nume, E1.Prenume, E1.NotaMedie;
- 2) E2.Nume, E2.Prenume, E2.NotaMedie;
- 3) P1.x, P1.y, P2.x, P2.y;
- 4) T1.A, T1.B, T1.C, T2.A, T2.B, T2.C;
- 5) T1.A.x, T1.A.y, T2.B.x, T2.B.y.

Evident, componenta **E1.Nume** este de tip **string**; componenta **P1.x** este de tip **real**; componenta **T1.A** este de tip **Punct**; componenta **T1.A.x** este de tip **real** ș.a.m.d.

Asupra componentelor datelor de tip *articol* se pot efectua toate operațiile admise de tipul câmpului respectiv. Programul ce urmează compară notele medii a doi elevi și afișează pe ecran numele și prenumele elevului cu nota medie mai bună. Se consideră că elevii au note medii diferite.

```
Program P85;
{ Date de tipul Elev }
type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
var E1, E2, E3 : Elev;
begin
    writeln('Dati datele primului elev:');
    write('Numele:');    readln(E1.Nume);
    write('Prenumele:'); readln(E1.Prenume);
    write('Nota medie:'); readln(E1.NotaMedie);

    writeln('Dati datele elevului al doilea:');
    write('Numele:');    readln(E2.Nume);
    write('Prenumele:'); readln(E2.Prenume);
    write('Nota medie:'); readln(E2.NotaMedie);

    if E1.NotaMedie > E2.NotaMedie then E3:=E1 else E3:=E2;

    writeln('Elevul cu media mai buna:');
    writeln(E3.Nume, ' ', E3.Prenume, ':', E3.NotaMedie : 5:2);
    readln;
end.
```

Orice tip de date **record** poate servi ca tip de bază pentru formarea altor tipuri structurate.

Exemplu:

```
type ListaElevilor = array [1..40] of Elev;
var LE : ListaElevilor;
```

Evident, notația **LE[i]** specifică elevul *i* din listă; notația **LE[i].Nume** specifică numele acestui elev, notația **LE[i].Prenume** specifică prenumele elevului respectiv ș.a.m.d. Programul ce urmează citește de la tastatură datele referitoare la *n* elevi și afișează pe ecran numele, prenumele și nota medie a celui mai bun elev. Se consideră că elevii au note medii diferite.

```

Program P86;
{ Tablou cu componente de tipul Elev }
type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
    ListaElev = array [1..40] of Elev;
var E : Elev;
    LE : ListaElev;
    n : 1..40;
    i : integer;
begin
    write('n='); readln(n);
    for i:=1 to n do
        begin
            writeln('Dati datele elevului ', i);
            write('Numele: ');      readln(LE[i].Nume);
            write('Prenumele: ');  readln(LE[i].Prenume);
            write('Nota Medie: '); readln(LE[i].NotaMedie);
        end;
        E.NotaMedie:=0;
        for i:=1 to n do
            if LE[i].NotaMedie > E.NotaMedie then E:=LE[i];
        writeln('Cel mai bun elev:');
        writeln(E.Nume, ' ', E.Prenume, ': ', E.NotaMedie : 5:2);
        readln;
    end.

```

În general, un tip de date *articol* se definește cu ajutorul diagramelor sintactice din figura 1.5. În completare la articolele cu un număr fix de câmpuri, limbajul PASCAL permite utilizarea articolelor cu variante. Aceste tipuri de date se studiază în cursurile avansate de informatică.

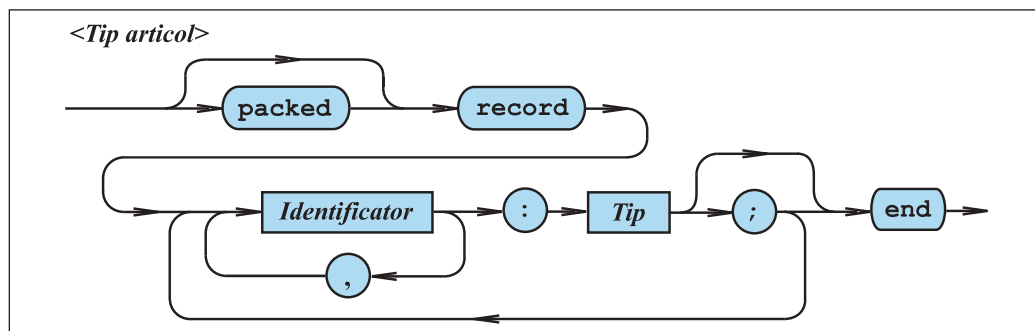


Fig. 1.5. Diagrama sintactică <Tip articol>

C++

În limbajul C++ un tip de date *articol* se definește printr-o structură gramaticală de forma:

```
struct [<Nume tip>] {  
     $T_1$     <Nume câmp 1>;  
     $T_2$     <Nume câmp 2>;  
    ...  
     $T_n$     <Nume câmp n>;  
}  
[lista variabile];
```

unde $T_1, T_2, \dots, T_n$ specifică tipul câmpurilor respective. Tipul unui nume de câmp este arbitrar, astfel un câmp poate să fie, la rândul lui, tot de tip *articol*. Prin urmare, se pot defini tipuri imbricate.

Exemple:

- 1)

```
struct Elev {  
    string Nume[20];  
    string Prenume[25];  
    float NotaMedie;  
};  
Elev E1, E2;
```
- 2)

```
struct Punct {  
    double x;    // coordonata x  
    double y;    // coordonata y  
};  
Punct P1, P2;
```
- 3)

```
struct Triunghi {  
    Punct A;    // vârful A  
    Punct B;    // vârful B  
    Punct C;    // vârful C  
};  
Triunghi T1, T2, T3;
```

Structura datelor din exemplele în studiu este prezentată în *figura 1.4**.

Fiind date două variabile de tip *înregistrare* de același tip, numele variabilelor pot apărea într-o instrucțiune de atribuire. Această atribuire înseamnă copierea tuturor câmpurilor din membrul drept în membrul stâng. De exemplu, pentru tipurile de date și variabilele declarate mai sus instrucțiunile

```
E1=E2;  
T2=T3;  
P2=P1
```

sunt corecte.

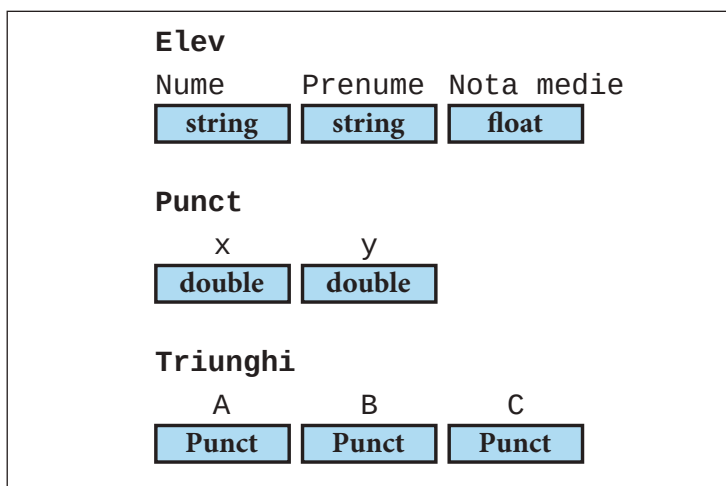


Fig. 1.4*. Structura datelor de tip Elev, Punct și Triunghi

Fiecare componentă a unei variabile de tip **struct** poate fi specificată explicit, prin numele variabilei și denumirile de câmpuri separate prin puncte.

Exemple:

- 1) E1.Nume, E1.Prenume, E1.NotaMedie
- 2) E2.Nume, E2.Prenume, E2.NotaMedie
- 3) P1.x, P1.y, P2.x, P2.y
- 4) T1.A, T1.B, T1.C, T2.A, T2.B, T2.C
- 5) T1.A.x, T1.A.y, T2.B.x, T2.B.y
- 6) V[3].Nume, v[3].Prenume, V[3].NotaMedie

Evident, componenta E1.Nume este de tip **string**; componenta P1.x este de tip **double**; componenta T1.A este de tip **Punct**; componenta T1.A.x este de tip **double** ș.a.m.d.

Asupra componentelor datelor de tip **struct** se pot efectua toate operațiile admise de tipul câmpului respectiv. Programul ce urmează compară notele medii a doi elevi și afișează pe ecran numele și prenumele elevului cu nota medie mai bună. Se consideră că elevii au note medii diferite.

```

//Program P85
// Date de tipul Elev
#include <iostream>
using namespace std;
int main()
{

```

```

struct Elev {
    char Nume[20];
    char Prenume[30];
    float NotaMedie;
};

Elev E1, E2, E3;
cout<<"Dati datele primului elev:"<<endl;
cout<<"Numele:"<<endl;      cin>>E1.Nume;
cout<<"Prenumele:"<<endl;    cin>>E1.Prenume;
cout<<"Nota medie:"<<endl;   cin>>E1.NotaMedie;
cout<<endl<<"Dati datele elevului al doilea:"<<endl;
cout<<"Numele:"<<endl;      cin>>E2.Nume;
cout<<"Prenumele:"<<endl;    cin>>E2.Prenume;
cout<<"Nota medie:"<<endl;   cin>>E2.NotaMedie;
if (E1.NotaMedie > E2.NotaMedie) E3=E1; else E3=E2;
cout<<"Elevul cu media mai bună:"<<endl;
cout<<E3.Nume<<' ' <<E3.Prenume<<' ' <<E3.NotaMedie;
return 0;
}

```

Orice tip de date **struct** poate servi ca tip de bază pentru formarea altor tipuri structurate.

Exemplu:

```

typedef Elev ListaElevilor[40];
ListaElevilor LE;

```

Evident, notația `LE[i]` specifică elevul `i` din listă; notația `LE[i].Nume` specifică numele acestui elev, notația `LE[i].Prenume` specifică prenumele elevului respectiv ș.a.m.d. Programul ce urmează citește de la tastatură datele referitoare la n elevi și afișează pe ecran numele, prenumele și nota medie a celui mai bun elev. Se consideră că elevii au note medii diferite.

```

//Program P86
// Tablou cu componente de tipul Elev
#include <iostream>
using namespace std;
int main()
{
    struct Elev {
        char Nume[20];
        char Prenume[30];
        float NotaMedie;
    };

    Elev E;
    int i;
    typedef Elev ListaElevilor[40];

```

```

ListaElevilor LE;
E.NotaMedie=0;
int n;
cout<<"n="; cin>>n;
for (i=0; i<n; i++)
{
    cout<<"Numele:"; cin>>LE[i].Nume;
    cout<<"Prenumele:";    cin>>LE[i].Prenume;
    cout<<"Nota medie:";    cin>>LE[i].NotaMedie;
};
for (i=0; i<n; i++)
    if (E.NotaMedie < LE[i].NotaMedie) E = LE[i];
cout << "Elevul cu nota medie mai mare:" << endl;
cout << E.Nume << ' ' << E.Prenume << ' ' << E.NotaMedie
<< endl;
return 0;
}

```

În general, un tip de date *înregistrare* se definește cu ajutorul diagramelor sintactice din *figura 1.5**. În completare la articolele cu un număr fix de câmpuri, limbajul C++ permite utilizarea înregistrărilor cu structură variabilă, definite prin tipul de date **union**. Aceste tipuri de date se studiază în cursurile avansate de informatică.

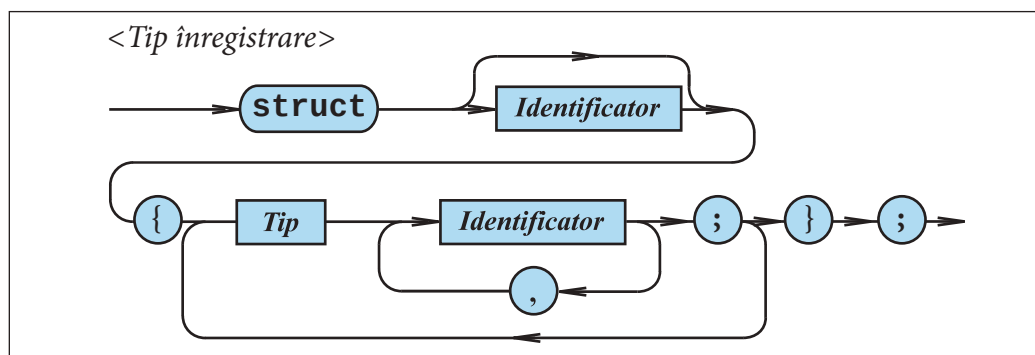


Fig. 1.5*. Diagrama sintactică <Tip articol>

Întrebări și exerciții

- ❶ Care este mulțimea de valori ale unui tip de date *articol*?
- ❷ Scrieți formulele metalingvistice pentru diagramele sintactice din *figurile 1.5/1.5**.
- ❸ Indicați pe diagramele sintactice din *figurile 1.5/1.5** drumurile care corespund definițiilor tipurilor de date *articol* din programul P85.
- ❹ EXERSEAZĂ! Definiți tipul de date *articol* destinat reprezentării pe calculator a următoarelor informații:
 - a) coordonatele carteziene ale unui punct pe plan;
 - b) coordonatele carteziene ale unui punct în spațiu;

- c) lista cărților din biblioteca personală: numele și prenumele autorului, titlul cărții, limba, editura, anul ediției, țara în care a fost editată;
- d) lista automobilelor dintr-un salon de vânzări auto: marca, tipul, culoarea caroseriei, capacitatea rezervorului, anul fabricației, prețul;
- e) lista pieselor muzicale din mediateca personală: numele și prenumele interpretului, titlul, genul de muzică, anul înregistrării;
- f) lista pasagerilor unor curse aeriene: numele, prenumele, numărul cursei (șir de o lungime de până la opt caractere), data (ziua, luna, anul) și ora decolării (ore, minute), tipul de avion, locul în avion (șir de o lungime de până la șase caractere);
- g) lista angajaților unei companii: numele, prenumele, postul ocupat, data angajării (ziua, luna, anul), vârsta, adresa domiciliului (localitatea, strada, casa, apartamentul).

⑥ APLICĂ! Se consideră următoarele declarații:

P A S C A L	C + +
<pre> type Elev = record Nume, Prenume : string; T, P:real; end; var E: Elev;</pre>	<pre> struct elev { char Nume[20]; char Prenume[25]; float T, P; }; Elev E;</pre>

În aceste declarații câmpul T este nota medie în semestrul de toamnă, iar câmpul P – nota medie a elevului în semestrul de primăvară. Scrieți instrucțiunea de atribuire ce calculează nota medie anuală M a elevului E.

⑥ REZOLVĂ! Se consideră următoarele tipuri de date

P A S C A L	C + +
<pre> type Data = record Ziua : 1..31; Luna : 1..12; Anul : integer; end; Persoana = record NumePrenume : string; DataNasterii : Data; end; ListaPersoane = array [1..50] of Persoana;</pre>	<pre> struct Data { int Ziua;Luna;Anul; }; struct Persoana { string NumePrenume; Data DataNasterii; }; Persoana ListaPersoane[50];</pre>

Elaborați un program care citește de pe tastatură datele referitoare la n persoane ($n \leq 50$) și afișează pe ecran:

- a) persoanele născute în ziua z a lunii;
- b) persoanele născute în luna l a anului;
- c) persoanele născute în anul a ;
- d) persoanele născute pe data $z.l.a$;
- e) persoana cea mai în vârstă;
- f) persoana cea mai tânără;

- g) vârsta fiecărei persoane în ani, luni, zile;
- h) lista persoanelor care au mai mult de v ani;
- i) lista persoanelor în ordine alfabetică;
- j) lista persoanelor ordonată conform datei nașterii;
- k) lista persoanelor de aceeași vârstă (născuți în același an).

- 7 Se consideră n puncte ($n \leq 30$) pe un plan euclidian. Fiecare punct i este definit prin coordonatele sale. Distanța dintre punctele i și j se calculează după formula

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Elaborați un program care afișează pe ecran punctele distanța dintre care este maximă.

- 8 Aria triunghiului este dată de formula lui Heron

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

unde p este semiperimetrul, iar a , b și c sunt lungimile laturilor respective. Utilizând tipurile de date `Punct` și `Triunghi` din paragraful în studiu, elaborați un program care citește de la tastatură informațiile referitoare la n triunghiuri ($n \leq 10$) și afișează pe ecran:

- a) aria fiecărui triunghi;
- b) coordonatele vârfurilor triunghiului cu aria maximă;
- c) coordonatele vârfurilor triunghiului cu aria minimă;
- d) informațiile referitoare la fiecare triunghi în ordinea creșterii ariilor.

1.5. Instrucțiunea `with`

NOTĂ

În limbajul C++ o astfel de instrucțiune nu există. Prin urmare, acest paragraf este destinat elevilor ce studiază doar limbajul PASCAL.

În limbajul PASCAL, componentele unei variabile de tip *articol* se specifică explicit prin numele variabilei și denumirile de câmpuri separate prin puncte.

De exemplu, în prezența declarațiilor

```
type Angajat = record
    NumePrenume : string;
    ZileLucrete : 1..31;
    PlataPeZi : real;
    PlataPeLuna : real;
end;
var A : Angajat;
```

componentele variabilei `A` se specifică prin `A.NumePrenume`, `A.ZileLucrete`, `A.PlataPeZi` și `A.PlataPeLuna`.

Întrucât numele `A` al variabilei de tip *articol* se repetă de mai multe ori, acest mod de referire a componentelor este, în anumite situații, incomod. Repetările obositoare pot fi evitate cu ajutorul instrucțiunii **with** (cu).

Sintaxa instrucțiunii în studiu este:

$\langle \text{Instrucțiune with} \rangle ::= \text{with } \langle \text{Variabilă} \rangle \{, \langle \text{Variabilă} \rangle\} \text{ do } \langle \text{Instrucțiune} \rangle$

Diagrama sintactică este prezentată în figura 1.6.

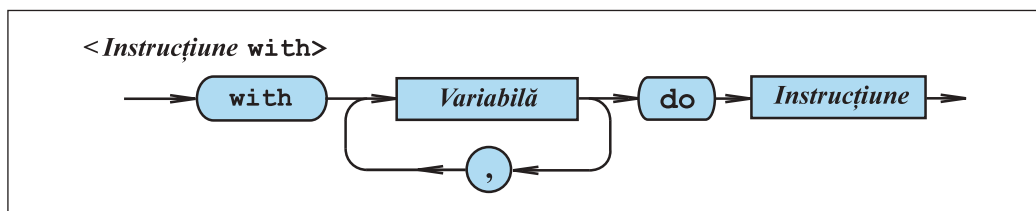


Fig. 1.6. Diagrama sintactică a instrucțiunii **with**

În interiorul unei instrucțiuni **with** componentele uneia sau a mai multe variabile de tip *articol* pot fi referite folosind numai numele câmpurilor respective.

Exemplu:

```
with A do PlataPeLuna:=PlataPeZi*ZileLucrate
```

Această instrucțiune este echivalentă cu următoarea:

```
A.PlataPeLuna:=A.PlataPeZi*A.ZileLucrate
```

Utilizarea instrucțiunii **with** necesită o atenție sporită din partea programatorului, care este obligat să specifice univoc componentele variabilelor de tip *articol*. În interiorul unei instrucțiuni **with**, la întâlnirea unui identificator, prima dată se testează dacă el poate fi interpretat ca un nume de câmp al articolului respectiv. Dacă da, identificatorul va fi interpretat ca atare, chiar dacă în acel moment este accesibilă și o variabilă având același nume.

Exemplu:

```
type Punct = record
    x : real;
    y : real;
end;
Segment = record
    A : Punct;
    B : Punct;
end;
var P : Punct;
    S : Segment;
    x : integer;
```

În cazul nostru identificatorul *x* poate să reprezinte fie variabila *x* de tip *integer*, fie câmpul *P.x* al articolului *P*.

În instrucțiunea

```
x:=1
```

identificatorul *x* se referă la variabila *x* de tip *integer*.

În instrucțiunea

```
with P do x:=1
```

identificatorul x se referă la câmpul $P.x$ al variabilei de tip articol P .

Întrucât variabila de tip articol S nu conține niciun câmp cu numele $S.x$, în instrucțiunea

```
with S do x:=1
```

identificatorul x va fi interpretat ca variabila x de tip `integer`.

O instrucțiune de forma

```
with  $v_1, v_2, \dots, v_n$  do <Instrucțiune>,
```

unde $v_1, v_2, \dots, v_n$ sunt variabile de tip *articol*, este echivalentă cu instrucțiunea

```
with  $v_1$  do  
with  $v_2$  do  
  { ... }  
with  $v_n$  do <Instrucțiune>.
```

Evident, componentele variabilelor $v_1, v_2, \dots, v_n$ trebuie specificate univoc prin denumirile câmpurilor respective.

De exemplu, pentru variabilele P și S , declarate mai sus, putem scrie:

```
with P, S do  
begin  
  x:=1.0;    { referire la P.x }  
  y:=1.0;  
  A.x:=0;    { referire la S.A.x }  
  A.y:=0;  
  B.x:=2.0; { referire la S.B.x }  
  B.y:=2.0;  
end;
```

În mod obișnuit, instrucțiunea **with** se utilizează numai în cazurile în care se ajunge la o reducere semnificativă a textului unui program.

Întrebări și exerciții

- ❶ Indicați pe diagrama sintactică din *figura 1.6* drumurile care corespund instrucțiunilor **with** din exemplele paragrafului în studiu.
- ❷ Care este destinația instrucțiunii **with**?
- ❸ APLICĂ! Utilizând instrucțiunea **with**, excludeți din programele P85 și P86 din paragraful precedent repetările de genul

```
E1.Nume, E1.Prenume, ...,  
LE[i].Nume, LE[i].Prenume.
```

- ④ Se consideră următoarele tipuri de date:

```
type Angajat = record
    NumePrenume : string;
    ZileLucrate : 1..31;
    PlataPeZi : real;
    PlataPeLuna : real;
end;
ListaDePlata = array [1..50] of Angajat;
```

Plata pe lună a fiecărui angajat se calculează înmulțind plata pe zi cu numărul de zile lucrate. Elaborați un program care:

- a) calculează plata pe lună a fiecărui angajat;
 - b) calculează salariul mediu al angajaților incluși în listă;
 - c) afișează pe ecran datele despre angajații cu plata lunară maximă;
 - d) afișează lista angajaților ordonată alfabetic;
 - e) afișează lista angajaților în ordinea creșterii plăților pe zi;
 - f) ordonează lista angajaților în ordinea creșterii plăților pe lună;
 - g) afișează lista angajaților în ordinea creșterii numărului de zile lucrate.
- ⑤ REZOLVĂ! Un cerc poate fi definit prin coordonatele x , y și raza r . Elaborați un program care citește de la tastatură datele referitoare la n cercuri ($n \leq 50$) și afișează pe ecran:
- a) coordonatele centrului și raza cercului cu aria maximă;
 - b) numărul de cercuri incluse în cercul cu raza maximă și coordonatele centrelor respective;
 - c) coordonatele centrului și raza cercului cu aria minimă;
 - d) numărul de cercuri în care este inclus cercul cu raza minimă și coordonatele centrelor respective.

1.6. Tipuri de date mulțime

Amintim că o mulțime este formată din obiecte fizice sau virtuale (ale gândirii) care au o proprietate comună. Obiectele din care este formată o mulțime se numesc *elementele mulțimii*. Elementele oricărei mulțimi sunt distincte, iar ordinea lor nu este importantă.

PASCAL

În limbajul PASCAL, un tip de date *mulțime* se definește în raport cu un tip de bază care trebuie să fie ordinal:

$\langle \text{Tip mulțime} \rangle ::= [\text{packed}] \text{ set of } \langle \text{Tip} \rangle$

Valorile unui tip de date **set** sunt mulțimi formate din valorile tipului de bază. Dacă tipul de bază are n valori, tipul *mulțime* va avea 2^n valori. În implementările limbajului valoarea lui n este limitată, de regulă $n \leq 256$.

În PASCAL o mulțime poate fi specificată enumerându-i-se elementele între parantezele pătrate „[” și „]”, care țin locul acoladelor din matematică.

Notăția `[]` reprezintă mulțimea vidă.

Exemple:

```

type Indice = 1..10;
      Zi = (L, Ma, Mi, J, V, S, D);
      MultimeIndicii = set of Indice;
      ZileDePrezenta = set of Zi;
var MI : MultimeIndicii;
      ZP : ZileDePrezenta;

```

Tipul ordinal *Indice* are $n = 10$ valori: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Tipul *MultimeIndicii* are $2^{10} = 1\,024$ de valori, și anume:

```

[], [1], [2], ..., [1, 2], [1, 3], ...,
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].

```

Prin urmare, variabila *MI* poate să aibă oricare dintre aceste valori, de exemplu :

```

MI := [1, 3].

```

Tipul ordinal *Zi* are $n = 7$ valori: L, Ma, Mi, J, V, S, D. Tipul *ZileDePrezenta* are $2^7 = 128$ de valori, și anume:

```

[], [L], [Ma], [Mi], ..., [L, Ma], [L, Mi], ...,
[L, Ma, Mi, J, V, S, D].

```

Variabila *ZP* poate să aibă oricare dintre aceste valori, de exemplu,

```

ZP := [L, Ma, Mi, V]

```

O valoare de tip *mulțime* poate fi specificată printr-un **constructor** (generator) de mulțime. Diagrama sintactică a unității gramaticale *<Constructor mulțime>* este prezentată în figura 1.7.

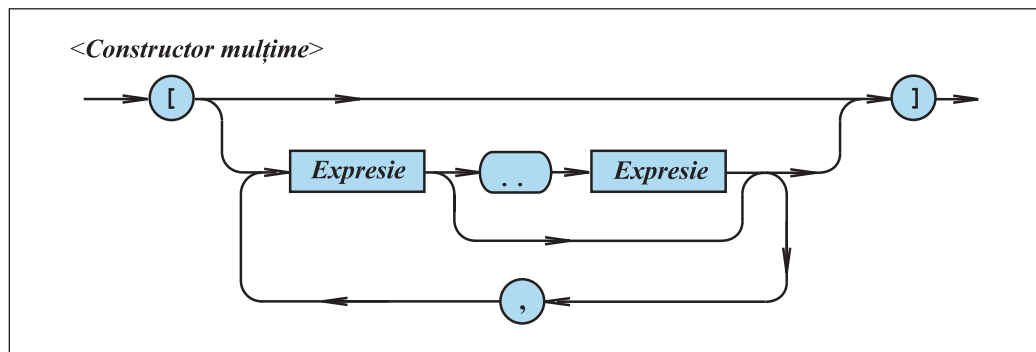


Fig. 1.7. Diagrama sintactică *<Constructor mulțime>*

Un constructor conține specificarea elementelor mulțimii, separate prin virgule și incluse între paranteze pătrate. Un element poate să fie o valoare concretă a tipului de bază sau un interval de forma:

```

<Expresie> .. <Expresie>.

```

Valorile expresiilor în studiu precizează limitele inferioare și superioare ale intervalului.

Exemple :

- 1) `[];`
- 2) `[1, 2, 3, 8];`
- 3) `[1..4, 8..10];`
- 4) `[i-k..i+k];`
- 5) `[L, Ma, V..D].`

Asupra valorilor unui tip de date *multime* se pot efectua operațiile uzuale:

- + reuniunea;
- * intersecția;
- diferența,

rezultatul fiind de tip *multime* și operațiile relaționale:

- = egalitatea;
- <> inegalitatea;
- <=, >= incluziunea;
- in** apartenența,

rezultatul fiind de tip *boolean*.

Programul ce urmează afișează pe ecran rezultatele operațiilor +, * și -, efectuate asupra valorilor de tip *MultimeIndicii*.

```
Program P87;
{ Date de tip MultimeIndicii }
type Indice = 1..10;
    MultimeIndicii = set of Indice;
var A, B, C : MultimeIndicii;
    i : integer;
begin
    A:= [1..5, 8];           { A contine 1, 2, 3, 4, 5, 8 }
    B:= [1..3, 9, 10];      { B contine 1, 2, 3, 9, 10 }
    C:= [];                 { C este o multime vida }
    C:=A+B;                 { C contine 1, 2, 3, 4, 5, 8, 9, 10 }

    writeln('Reuniune');
    for i:=1 to 10 do
        if i in C then write(i:3);
    writeln;

    C:=A*B;                 { C contine 1, 2, 3 }
    writeln('Intersectie');
    for i:=1 to 10 do
        if i in C then write(i:3);
    writeln;
```

```

C:=A-B;           { C contine 4, 5, 8 }
writeln('Diferenta');
for i:=1 to 10 do
  if i in C then write(i:3);
writeln;
readln;
end.

```

Spre deosebire de tablouri și articole, componentele cărora pot fi referite direct, respectiv prin indicii și denumiri de câmpuri, elementele unei mulțimi nu pot fi referite. Se admite numai verificarea apartenenței elementului la o mulțime (operația relațională **in**). În pofida acestui fapt, utilizarea tipurilor de date *mulțime* mărește viteza de execuție și îmbunătățește lizibilitatea programelor PASCAL. De exemplu, instrucțiunea:

```

if (c='A') or (c='E') or (c='I') or (c='O') or (c='U') then ...

```

poate fi înlocuită cu o instrucțiune mai simplă:

```

if c in ['A','E','I','O','U'] then ...

```

ȘTIAI CĂ?

Eratostene a fost un matematician, geograf, poet, astronom și muzician grec, care a trăit între anii 276 și 194 î.Hr. A fost primul om care a calculat circumferința pământului, a calculat înclinația axei de rotație a pământului, a formulat conceptul de an bisect, a desenat una dintre primele hărți ale lumii. Eratostene a inventat un algoritm foarte eficient de calcul al tuturor numerelor prime ce nu depășesc un număr dat.

Sursa: <https://www.britannica.com>



Un alt exemplu sugestiv este utilizarea tipurilor de date *mulțime* în calcularea numerelor prime mai mici decât un număr natural dat n . Pentru aceasta se folosește algoritmul *Ciurul (sita) lui Eratostene*:

- 1) în sită se depun numerele 2, 3, 4, ..., n ;
- 2) din sită se extrage cel mai mic număr i ;
- 3) numărul extras se include în mulțimea numerelor prime;
- 4) din sită se elimină toți multiplii m ai numărului i ;
- 5) procesul se încheie când sita s-a golit.

```

Program P88;
{ Ciurul (sita) lui Eratostene }
const n = 50;

```

```

type MultimeDeNumere = set of 1..n;
var Sita, NumerePrime : MultimeDeNumere;
    i, m : integer;
begin
  {1} Sita:= [2..n];
      NumerePrime:=[];
      i:=2;
      repeat
  {2}   while not (i in Sita) do i:=succ(i);
  {3}   NumerePrime:=NumerePrime+[i];
        write(i:4);
        m:=i;
  {4}   while m<=n do
        begin Sita:=Sita-[m]; m:=m+i; end;
  {5} until Sita=[];
        writeln;
        readln;
end.

```

Correspondența dintre punctele algoritmului și instrucțiunile care le exprimă este indicată în comentariile din partea stângă a liniilor de program.

C++

În limbajul C++ mulțimile pot fi reprezentate prin tablouri unidimensionale. De obicei, componentele respective sunt declarate ca aparținând unor tipuri ordinale de date, adică **int**, **bool**, **char** și **enum**.

Exemple:

1) mulțimile A formate din numere întregi ce conțin până la 100 de elemente pot fi reprezentate prin tabloul **int** $A[100]$;

2) mulțimile B formate din cele 26 de caractere ale alfabetului latin pot fi reprezentate prin tabloul **char** $B[26]$;

3) Mulțimile Z formate din valorile tipului de date **enum** Z_i {L, Ma, Mi, J, V, S, D} pot fi reprezentate prin tabloul Z_i $Z[7]$.

Evident, în afară de tabloul propriu-zis, în care sunt memorate elementele mulțimii supuse prelucrării, va trebui să mai declarăm o variabilă întreagă n în care să memorăm numărul curent de elemente ale mulțimii respective.

În cazul mulțimilor A , B și Z descrise mai sus, pot fi declarate variabile n_A , n_B și n_Z , de exemplu: **int** n_A , n_B , n_Z .

Prezentăm în continuare un program care citește de la tastatură mulțimile A și B , formate din cel mult 20 de numere întregi, și afișează pe ecran intersecția acestora, adică mulțimea $C = A \cap B$.

Amintim că mulțimea C trebuie să conțină doar elementele care apar atât în mulțimea A , cât și în mulțimea B .

```

// Program P87
// Reprezentarea multimilor prin tablouri
// Intersectia multimilor A si B

```

```

#include <iostream>
using namespace std;
int main ()
{
    int A[20], B[20], C[20],
        nA, nB, nC,
        i, j, k; // contoare cicluri
    bool Gasit;   // fanion: elementul multimii A
                  // se regaseste in multimea B
    cout << "Dati numarul de elemente ale multimii A: ";
    cin >> nA;
    cout << "Dati elementele multimii A:" << endl;
    for (i=0; i<nA; i++) cin >> A[i];
    cout << "Dati numarul de elemente ale multimii B: ";
    cin >> nB;
    cout << "Dati elementele multimii B:" << endl;
    for (j=0; j<nB; j++) cin >> B[j];
    nC=0; // initial multimea C este vida
    for (i=0; i<nA; i++)
    {
        Gasit = false;
        for (j=0; j<nB; j++) if (A[i]==B[j]) Gasit = true;
        if (Gasit==true) { C[nC] = A[i]; nC++; };
    }
    cout << "Numarul de elemente ale multimii C = " << nC << endl;
    cout << "Multimea C:" << endl;
    for (k=0; k<nC; k++) cout << C[k] << " ";
    cout << endl;
    return 0;
}

```

O altă metodă de reprezentare a mulțimilor constă în utilizarea așa-numitor **vectori caracteristici**. În această metodă se presupune că există o mulțime universală (*universum*), notată prin U , ce conține toate elementele din care pot fi formate mulțimile de prelucrat.

De exemplu, dacă se preconizează prelucrarea mulțimilor, elementele cărora sunt numere naturale mai mici ca 100, atunci $U = \{0, 1, 2, 3, \dots, 99\}$. Într-un mod similar, dacă se presupune prelucrarea mulțimilor, elementele cărora sunt literele mari ale alfabetului latin, atunci $U = \{A, B, C, \dots, Z\}$.

Presupunem că mulțimea universală conține n elemente:

$$U = \{u_1, u_2, u_3, \dots, u_n\},$$

iar mulțimea de prelucrat conține m elemente:

$$A = \{a_1, a_2, a_3, \dots, a_m\}.$$

Pe calculator, această mulțime poate fi reprezentată prin vectorul caracteristic

$$V = (v_1, v_2, v_3, \dots, v_i, \dots, v_n),$$

unde

$$v_i = \begin{cases} 1, & \text{dacă } u_i \in A; \\ 0, & \text{în caz contrar.} \end{cases}$$

Cu alte cuvinte, componenta i a vectorului caracteristic V are valoarea 1 dacă elementul respectiv din mulțimea universală aparține mulțimii A și valoarea 0 în caz contrar.

De exemplu, dacă $U = \{1, 2, 3, 4, 5\}$ și $A = \{2, 4\}$, atunci vectorul caracteristic:

$$V = (0, 1, 0, 1, 0).$$

Într-un mod similar, dacă $U = \{a, b, c, d, e, f\}$ și $B = \{a, c, e, f\}$, atunci vectorul caracteristic:

$$V = (1, 0, 1, 0, 1, 1).$$

Din exemplele de mai sus se observă că numărul de componente ale vectorului caracteristic V al unei mulțimi A coincide cu numărul de elemente n ale mulțimii universale U . Evident, numărul de componente nenule ale vectorului caracteristic V este egal cu numărul de elemente m ale mulțimii A .

Un exemplu sugestiv de utilizare a vectorilor caracteristici pentru reprezentarea mulțimilor de prelucrat este calcularea numerelor prime mai mici decât un număr natural dat n . Pentru aceasta se folosește algoritmul *Ciurul (sita) lui Eratostene*:

- 1) în sită se depun numerele 2, 3, 4, ..., n ;
- 2) din sită se extrage cel mai mic număr i ;
- 3) numărul extras se include în mulțimea numerelor prime;
- 4) din sită se elimină toți multiplii m ai numărului i ;
- 5) procesul se încheie când sita s-a golit.

```
// Program P88
// Ciurul (sita) lui Eratostene
// Reprezentarea multimilor prin vectori caracteristici
// Universum U = {0, 1, 2, ..., n}
#include <iostream>
using namespace std;
int main()
{
    const int nmax = 51;
    int n, S[nmax], // Vectorul caracteristic Sita
        P[nmax],   // Vectorul caracteristic Numere prime
        i,         // Numarul extras din Sita
        j, k;      // Contoare ciclu
    cout << "Dati n= "; cin >> n;
    // Depunem numerele 2, 3, 4, .., n in Sita
    S[0] = 0; S[1] = 0;
    for (j = 2; j <= n; j++) S[j] = 1;
    // Zerografiem multimea numerelor prime
    for (j = 0; j <= n; j++) P[j] = 0;
    // Extragem din sita cel mai mic numar i
```

```

for (i = 2; i <= n; i++)
{
    if (S[i] != 0)
    {
        P[i] = 1; // includem i in Numere prime
        S[i] = 0; // excludem i din Sita
        // Eliminam din Sita toti multiplii lui i
        for (j = i; j <= n; j++) if ((j % i) == 0) S[j] = 0;
    }
}
cout << "Numerele prime:" << endl;
for (j = 2; j <= n; j++) if (P[j] == 1) cout << j << ' ';
cout << endl;
return 0;
}

```

Întrebări și exerciții

- ❶ OBSERVĂ! (PASCAL) Enumerați valorile posibile ale variabilelor din declarațiile ce urmează:

```

var V : set of 'A'..'C';
    S : set of (A, B, C);
    I : set of '1'..'2';
    J : set of 1..2;

```

- ❷ OBSERVĂ! (PASCAL) Comentăți următorul program:

```

Program P89;
{ Eroare }
type Multime = set of integer;
var M : Multime;
    i : integer;
begin
    M := [1, 8, 13];
    for i := 1 to MaxInt do
        if i in M then writeln(i);
end.

```

- ❸ EXERSEAZĂ! (PASCAL) Se consideră următoarele declarații:

```

type Culoare = (Galben, Verde, Albastru, Violet);
    Nuanta = set of Culoare;
var NT : Nuanta;

```

Listați valorile posibile ale variabilei NT.

- ❹ EXERSEAZĂ! (PASCAL) Scrieți formula metalingvistică care corespunde diagramei sintactice <Constructor multime> din figura 1.7.

- 5 EXERSEAZĂ! (PASCAL) Se consideră tipul de date `MultimeIndicii` din paragraful în studiu. Precizați mulțimile specificate de constructorii ce urmează:
- | | |
|---------------------------------|----------------------------------|
| a) <code>[];</code> | f) <code>[4..3];</code> |
| b) <code>[1..10];</code> | g) <code>[1..3, 7..6, 9];</code> |
| c) <code>[1..3, 9..10];</code> | h) <code>[4-2..7+1];</code> |
| d) <code>[1+1, 4..7, 9];</code> | i) <code>[7-5..4+4];</code> |
| e) <code>[3, 7..9];</code> | j) <code>[6, 9, 1..2].</code> |
- 6 EXERSEAZĂ! (C++) Se consideră mulțimea universală $U = \{L, Ma, Mi, J, V, S, D\}$, care reprezintă zilele săptămânii. Scrieți vectorii caracteristici ai următoarelor mulțimi:
- $A = \{S, D\}$ – zilele de odihnă ale săptămânii;
 $B = \{L, Ma, Mi, J, V\}$ – zilele de lucru ale săptămânii;
 $C = \{L, Mi, J\}$ – zilele în care au loc consultații online la disciplina școlară Informatica;
 $D = \{Ma, Mi\}$ – zilele în care își desfășoară activitatea cercurile artistice ale liceului;
 $E = \{Ma, J, V\}$ – zilele în care își desfășoară activitatea secțiile sportive ale liceului;
 $F = \{L, V\}$ – zilele în care biblioteca liceului organizează expoziții de carte.
- 7 EXERSEAZĂ! (C++) Utilizând mulțimile U, A, B, C, D, E, F din exercițiul precedent, scrieți vectorii caracteristici ai mulțimilor ce conțin:
- G – zilele săptămânii în care își desfășoară activitatea atât cercurile artistice, cât și secțiile sportive ale liceului;
 H – zilele săptămânii în care își desfășoară activitatea fie cercurile artistice, fie secțiile sportive ale liceului;
 I – zilele săptămânii în care atât cercurile artistice, cât și secțiile sportive nu activează.
- Elaborați un program care calculează și afișează aceste mulțimi pe ecran.
- 8 PROGRAMEAZĂ! Scrieți un program care citește de la tastatură elementele mulțimilor A și B și afișează pe ecran:
- a) intersecția mulțimilor A și B ;
b) reuniunea mulțimilor A și B ;
c) diferența mulțimilor A și B .
- Mulțimile A și B sunt formate din numere întregi.
- 9 PROGRAMEAZĂ! Scrieți un program care citește de la tastatură elementele mulțimilor A și B și afișează pe ecran:
- a) intersecția mulțimilor A și B ;
b) reuniunea mulțimilor A și B ;
c) diferența mulțimilor A și B ;
- Mulțimile A și B sunt formate din literele mari ale alfabetului latin.
- 10 Elaborați un program care afișează pe ecran toate submulțimile mulțimii $\{1, 2, 3, 4\}$.
- 11 Elaborați un program care afișează pe ecran toate submulțimile mulțimii $\{A, B, C, D\}$.
- 12 Se consideră șiruri de caractere formate din literele mari și mici ale alfabetului latin, în care cuvintele sunt separate prin spațiu, punct, virgulă, punct și virgulă,

semnul exclamării sau semnul întrebării. Elaborați un program care afișează pe ecran cuvintele din șirurile citite de la tastatură.

- 13 Se consideră șiruri de caractere formate din literele mari și mici ale alfabetului latin. Elaborați un program care afișează pe ecran numărul de vocale din șirurile citite de la tastatură.

- 14 Elaborați un program care citește de la tastatură două șiruri de caractere și afișează pe ecran:

a) caracterele care se întâlnesc cel puțin în unul dintre șiruri;

b) caracterele care apar în ambele șiruri;

c) caracterele care apar în primul și nu apar în șirul al doilea.

- 15 Se consideră că numele unei persoane este scris corect dacă el este format din literele mari și cele mici ale alfabetului latin, începe cu literă mare, iar literele ce urmează sunt mici. Scrieți un program care verifică dacă șirul de caractere citit de la tastatură reprezintă un nume corect de persoană. În cazul unui nume corect, la ecran se va afișa mesajul **CORECT**, iar în caz contrar – **INCORECT**.

- 16 În implementările actuale ale limbajului numărul valorilor tipului de bază al unui tip *multime* este limitat, obișnuit $n \leq 256$. În consecință, programul **P88** nu poate calcula numere prime mai mari decât n . Elaborați un program pentru calcularea numerelor prime din intervalul 8, ..., 10 000.

Indicație: Sita din algoritmul lui Eratostene poate fi reprezentată printr-un tablou, componentele căruia sunt mulțimi.

- 17 **STUDIUL DE CAZ.** (C++) Scrieți un program care calculează numerele prime cu ajutorul algoritmului *Ciurul (sita) lui Eratostene* reprezentând mulțimile *Sita* și *Numere prime* nu prin vectori caracteristici, ci prin tablouri ce conțin elementele propriu-zise ale acestora. Comparați complexitatea programului scris cu programul care utilizează vectori caracteristici.

- 18 **CREEAZĂ!** Se consideră posturile de televiziune A și B . În anumite zile ale săptămânii, nu neapărat aceleași, fiecare dintre aceste posturi nu difuzează filme artistice. De exemplu, Postul A nu difuzează filme artistice în zilele de *luni* și *vineri*, iar postul B – în zilele de *marți* și *duminică*.

Scrieți un program care, pentru fiecare din posturile A și B , citește de la tastatură zilele săptămânii în care nu sunt difuzate filme artistice și afișează pe ecran zilele în care spectatorul are posibilitatea să vizioneze un film artistic pe cel puțin unul din posturi. Zilele săptămânii vor fi citite de la tastatură și afișate pe ecran în forma uzuală, adică *Luni*, *Marți*, *Miercuri* ș.a.m.d., fără a folosi numerele lor de ordine sau abrevieri.

Indicație: Utilizați mulțimi elementele cărora sunt de tipul ordinal de date *enumerare*.

- 19 **DESCOPERĂ!** (C++) Este cunoscut faptul că mulțimile pot fi reprezentate prin tablouri unidimensionale, componentele cărora conțin elementele propriu-zise ale acestor mulțimi. Totodată, cu fiecare tablou trebuie să fie asociată o mărime întreagă ce conține numărul curent de elemente ale mulțimii respective. De exemplu, în cazul mulțimii A , formată din cel mult 20 de numere întregi, cu tabloul `int A[20]` se asociază variabila nA .

Pentru a face programele mai intuitive, mulțimile pot fi reprezentate și cu ajutorul tipului de date *articol*, care are forma `struct Multime {int E[nmax], nE;}`. În acest tip de date tabloul E este destinat stocării elementelor, iar variabila nE – stocării numărului curent de elemente ale mulțimii respective.

Rescrieți programele C++ din acest paragraf utilizând pentru reprezentarea mulțimilor date de tipul *articol*.

- ❷ **ÎNVĂȚĂ SĂ ÎNVETI! (C++)** Biblioteca standard a limbajului C++ conține containerul special *Set*, destinat reprezentării și prelucrării datelor de tip *mulțime*. Recomandăm elevilor pasionați de programare să studieze în mod individual sau în echipă mijloacele oferite de acest container. Încercați să transcrieți programele din acest paragraf cu ajutorul funcțiilor definite în acest container.

1.7. Generalități despre fișiere

Până în prezent, programele elaborate de noi citeau datele de prelucrat de la tastatură și afișau rezultatele prelucrărilor pe ecran. Evident, datele introduse de la tastatură se memorau în variabilele din programele respective. Însă, odată cu terminarea execuției oricărui program, valorile variabilelor din acest program se pierd, fapt care cere introducerea repetată a datelor de intrare la o nouă lansare în execuție a acestuia. În cazul unor date de intrare voluminoase, introducerile repetate de date cer un volum foarte mare de muncă.

Totodată, din experiența acumulată de noi în procesul prelucrării textelor, imaginilor, secvențelor audio și video, cunoaștem deja că datele de prelucrat și, desigur, rezultatele prelucrărilor, trebuie păstrate în fișiere. Amintim, fișierul reprezintă o colecție de date ce are un nume și care se păstrează pe un suport extern de informație, de exemplu, pe discurile magnetice sau discurile optice, pe memoriile de tip *flash* ș.a.

Pentru a putea prelucra informațiile din fișierele de pe suporturile externe de informații, limbajele moderne de programare conțin tipuri structurate de date cu denumirea generică de *fișier*. În limbajele PASCAL și C++, prin **fișier** se înțelege o structură de date care constă dintr-o secvență de componente (înregistrări). Numărul componentelor din secvență nu este fixat, însă sfârșitul secvenței este indicat de un simbol special, notat *EOF* (*End of File* – sfârșit de fișier). Fișierul care nu conține nicio componentă se numește *fișier vid*.

În ansamblu, utilizarea fișierelor are următoarele avantaje:

- se asigură păstrarea de lungă durată atât a datelor de intrare, cât și a celor de ieșire, utilizatorul având posibilitatea să revină oricând la ele după terminarea procesului de execuție a programelor respective;
- programele devin universale, întrucât unul și același program poate fi utilizat fără modificări pentru prelucrarea datelor din diverse fișiere externe;
- prelucrările complexe pot fi efectuate prin divizarea sarcinilor între diferite programe mai simple, întrucât datele de ieșire ale unui program pot fi utilizate ca date de intrare pentru un alt program.

ATENȚIE!

Mijloacele destinate prelucrării fișierelor depind în mare măsură de specificul sistemelor de operare și mediile de programare utilizate. În consecință, rezultatele furnizate de programele ce prelucrează fișiere ar putea să difere de la un calculator la altul.

Prin urmare, elevii sunt încurajați să studieze mijloacele destinate prelucrării fișierelor prin **metoda experimentării pe calculator**. Se recomandă ca fiecare din elevi să scrie și să lanseze în execuție mici programe ce operează cu datele de pe suporturile externe de informații.

PASCAL

În limbajul PASCAL, un tip de date *fișier* se definește printr-o declarație de forma:

$\langle \text{Tip fișier} \rangle ::= [\text{packed}] \text{ file of } \langle \text{Tip} \rangle;$

unde $\langle \text{Tip} \rangle$ este tipul de bază, adică tipul componentelor fișierelor respective. Tipul de bază este un tip arbitrar, exceptând tipul *fișier* (nu există „fișier de fișiere”).

Exemple:

```
1) type FisierNumere = file of integer;
   var FN : FisierNumere;
       n : integer;
```

```
2) type FisierCaractere = file of char;
   var FC : FisierCaractere;
       c : char;
```

```
3) type Elev = record
       Nume : string;
       Prenume : string;
       NotaMedie : real;
     end;
   FisierElevi = file of Elev;
   var FE : FisierElevi;
       E : Elev;
```

Structura datelor în studiu este prezentată în *figura 1.8*. Subliniem faptul că elementul *EOF*, care indică sfârșitul secvenței, nu este o componentă a fișierului.

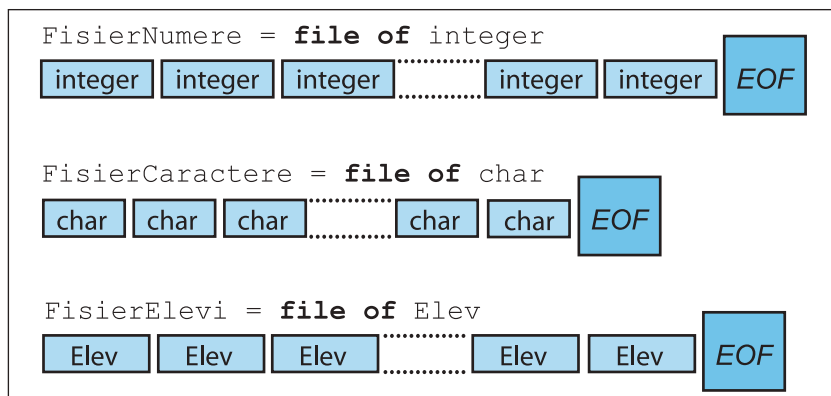


Fig. 1.8. Structura datelor de tip FisierNumere, FisierCaractere și FisierElevi

Variabilele FN, FC, FE ș.a. de tip *fișier* se numesc **fișiere logice**, **fișiere PASCAL** sau, pur și simplu, **fișiere**. Spre deosebire de celelalte tipuri de date, valorile cărora se păstrează în memoria internă a calculatorului, datele fișierelor PASCAL se păstrează pe suporturile de informație ale echipamentelor periferice (discuri și benzi magnetice, discuri optice, hârtia imprimantei sau a dispozitivului de citit

documente ș.a.). Informația pe suporturile în studiu este organizată în formă de **fișiere externe** în conformitate cu cerințele sistemului de operare. Prin urmare, înainte de a fi utilizată, o variabilă de tip fișier trebuie asociată cu un fișier extern. Metodele de asociere sunt specifice implementării limbajului și sistemului de operare al calculatorului-gazdă.

În limbajul-standard asocierea se realizează prin includerea variabilelor de tip fișier ca argumente în antetul de program.

În Turbo PASCAL asocierea unei variabile de tip *fișier* f cu un fișier extern se realizează prin apelul de procedură

```
assign( $f$ ,  $s$ );
```

unde s este o expresie de tip **string** care specifică calea de acces și numele fișierului extern.

Exemple:

```
1) assign(FN, 'A:\REZULTAT\R.DAT')
```

– fișierul FN se asociază cu fișierul extern R.DAT din directorul REZULTAT de pe discul A.

```
2) assign(FC, 'C:\A.CHR')
```

– fișierul FC se asociază cu fișierul A.CHR din directorul-rădăcină al discului C.

```
3) write('Dati un nume de fisier:');  
    readln(str);  
    assign(FE, str);
```

– fișierul FE se asociază cu un fișier extern numele căruia este citit de la tastatură și depus în variabila de tip **string** str .

După executarea instrucțiunii `assign( $f$ ,  $s$ )`, toate operațiile referitoare la fișierul PASCAL f se vor efectua asupra fișierului extern s . Accentuăm faptul că dacă utilizatorul nu indică numele complet al fișierului extern (directorul în care se află și numele propriu-zis al fișierului), el va fi creat/căutat în directorul implicit al mediului de dezvoltare a programelor PASCAL.

Cele mai uzuale operații asupra unui fișier sunt citirea și scrierea unei componente.

Citirea unei componente se realizează printr-un apel de forma

```
read( $f$ ,  $v$ ),
```

unde v este o variabilă declarată cu tipul de bază al fișierului f .

Scrierea unei componente se realizează printr-un apel de forma:

```
write( $f$ ,  $e$ ),
```

unde e este o expresie asociată cu tipul de bază al fișierului f .

Exemple:

- 1) `read(FN, n);`
- 2) `write(FC, c);`
- 3) `read(FE, E).`

După **tipul operațiilor permise** asupra componentelor, fișierele se clasifică în:

- fișiere de intrare (este permisă numai citirea);
- fișiere de ieșire (este permisă numai scrierea);
- fișiere de actualizare (sunt permise atât scrierea, cât și citirea).

După **modul de acces la componente**, fișierele se clasifică în:

- fișiere cu acces secvențial sau secvențiale (accesul la componenta i este permis după ce s-a citit/scriș componenta $i - 1$);
- fișiere cu acces aleatoriu sau direct (orice componentă se poate referi direct prin numărul ei de ordine i în fișier).

Menționăm că în limbajul-standard sunt permise numai fișiere secvențiale de intrare sau ieșire.

Tipul fișierului (de intrare, ieșire sau de actualizare) și modul de acces (secvențial sau direct) se stabilesc printr-o operație de validare, numită **deschidere a fișierului**. Pentru aceasta, în limbajul-standard se utilizează următoarele proceduri:

`reset(f)` – pregătește un fișier existent pentru citire;

`rewrite(f)` – creează un fișier vid și îl pregătește pentru scriere.

Când prelucrarea componentelor se termină, fișierul trebuie închis. La **închiderea** unui fișier sistemul de operare înscrie elementul *EOF*, înregistrează fișierul extern nou-creat în directorul respectiv ș.a.m.d.

În limbajul-standard se consideră că fișierele vor fi închise implicit la terminarea execuției programului respectiv. În Turbo PASCAL fișierul f se închide prin apelul de procedură `close(f)`.

În concluzie, prezentăm ordinea în care trebuie apelate procedurile destinate prelucrării datelor de tip *fișier*:

1. **Asocierea** fișierului PASCAL f cu fișierul extern s : `assign(f, s)`.

2. **Deschiderea** fișierului f pentru citire sau scriere: `reset(f)`, respectiv, `rewrite(f)`.

3. **Citirea/scrierea** unei componente a fișierului f : `read(f, v)`, respectiv, `write(f, e)`.

4. **Închiderea** fișierului f : `close(f)`.

După închiderea fișierului, variabila f poate fi asociată cu un alt fișier extern.

Întrucât valorile variabilelor de tip fișier se păstrează pe suporturile externe de informație, în PASCAL **atribuirile de fișiere sunt interzise**.

C++

În limbajul C++ un fișier este alcătuit din înregistrări (componente), care pot avea dimensiuni fixe sau variabile. Numărul înregistrărilor dintr-un fișier este limitat de capacitatea de memorare a suportului fizic folosit.

După **tipul operațiilor permise asupra componentelor**, fișierele se clasifică în:

- fișiere de intrare (este permisă numai citirea);
- fișiere de ieșire (este permisă numai scrierea);
- fișiere de actualizare (sunt permise atât scrierea, cât și citirea).

După **modul de acces la componente**, fișierele se clasifică în:

- fișiere cu acces secvențial (înregistrările pot fi prelucrate numai în ordinea în care ele sunt stocate în fișier);
- fișiere cu acces direct (componentele pot fi prelucrate în orice ordine). În acest caz, înainte de fiecare operație de citire/scriere se indică informația necesară pentru a selecta componenta ce urmează a fi prelucrată.

După **modul de interpretare a conținuturilor**, fișierele se clasifică în:

- fișiere text care conțin numai caractere structurate pe linii;
- fișiere binare în care informația este văzută ca o colecție de octeți.

Asupra fișierelor pot fi efectuate următoarele **operații**:

- deschiderea fișierului;
- citirea datelor din fișier/scrierea datelor în fișier;
- închiderea fișierului.

În limbajul C++ prelucrarea datelor din fișierele externe se realizează cu ajutorul fluxurilor (*stream*-urilor).

Fluxurile sunt obiecte care transportă și formează șiruri de octeți. Pentru exemplificare, amintim fluxurile pe care le cunoaștem deja:

cin – fluxul destinat citirii datelor de pe suportul extern de informație al dispozitivului standard de intrare, de obicei, tastatura;

cout – fluxul destinat scrierii datelor pe suportul extern de informație al dispozitivului standard de ieșire, de obicei, ecranul.

În general, programatorul poate crea și fluxuri proprii. Astfel de fluxuri trebuie asociate cu fișierele de pe dispozitivele externe, de exemplu, cu fișierele de pe discurile magnetice sau din memoriile de tip *flash*. După asociere, operația de includere a datelor într-un flux are ca efect scrierea acestora în fișierul asociat, iar extragerea datelor dintr-un flux – citirea acestora din fișierul respectiv.

Amintim că includerea datelor într-un flux se efectuează cu ajutorul operatorului <<, iar extragerea datelor – cu ajutorul operatorului >>.

Pentru a avea acces la funcțiile destinate lucrului cu fluxuri și fișiere, în programele C++ trebuie inclusă directiva:

```
#include <fstream>
```

Crearea unui flux de intrare și asocierea lui cu un fișier extern se efectuează cu ajutorul următoarei construcții gramaticale:

```
ifstream <Nume flux de intrare> ("<Nume fișier extern>") ;
```

Evident, fișierul extern din care vor fi citite datele trebuie să existe pe suportul de informații până la lansarea programului respectiv în execuție. În caz contrar, în procesul executării programului, în momentul asocierii fluxului de intrare cu fișierul extern se va declanșa o eroare.

Crearea unui flux de ieșire și asocierea lui cu un fișier extern se efectuează cu ajutorul următoarei construcții gramaticale:

```
ofstream <Nume flux de ieșire> ("<Nume fișier extern>") ;
```

Dacă în momentul asocierii fluxului de ieșire cu fișierul extern ultimul încă nu există, el va fi creat în mod automat. Dacă fișierul extern există deja, toate înregistrările din el vor fi șterse.

Exemple:

1) `ifstream FI("C:\REZULTAT\R.DAT");`

- crearea fluxului de intrare **FI** și asocierea lui cu fișierul extern **R.DAT** din directorul **REZULTAT** de pe discul **C:**.

2) `ofstream FE("D:\D.CHR");`

- crearea fluxului de ieșire **FE** și asocierea lui cu fișierul extern **D.CHR** din directorul-rădăcină al discului **D:**.

3) `cout << ("Dati un nume de fisier:");
cin << (Str);
ofstream FE(Str);`

- crearea fluxului de ieșire **FE**. Fluxul **FE** se asociază cu un fișier extern numele căruia este citit de la tastatură și depus în variabila de tip **string** **Str**.

Accentuăm faptul că în ultimul exemplu numele fișierului extern nu este cunoscut la momentul scrierii programului, el fiind citit de la tastatură doar în momentul execuției acestuia. O astfel de abordare permite elaborarea de programe ce pot prelucra datele nu doar dintr-un anumit fișier extern, ci din oricare alt fișier indicat de utilizator.

Variabilele (fluxurile) **FI** și **FE** din exemplele de mai sus se mai numesc **fișiere logice** sau, pur și simplu, **fișiere**. Spre deosebire de celelalte tipuri de date, valorile cărora se păstrează în memoria internă a calculatorului, datele fișierelor se păstrează pe suporturile de informație ale echipamentelor periferice (discuri și benzi magnetice, discuri optice, memorii *flash* ș.a.). Informația pe suporturile în studiu este organizată în formă de **fișiere externe** în conformitate cu cerințele sistemului de operare.

Accentuăm faptul că dacă utilizatorul nu indică numele complet al fișierului extern (directorul în care se află și numele propriu-zis al fișierului), el va fi creat/căutat în directorul implicit al mediului de dezvoltare a programelor C++.

Închiderea fișierelor asociate fluxurilor se efectuează prin apeluri de forma:

```
<Nume flux>.close();
```

Exemple:

```
FI.close(); FE.close();
```

După ce un fișier a fost închis, în el nu mai pot fi scrise și din el nu mai pot fi citite niciun fel de date. Însă aceste operații devin posibile după o nouă deschidere a fișierului în cauză. Cu alte cuvinte, scrierea sau citirea datelor, după caz, este posibilă doar în perioada în care fișierul respectiv este deschis.

Evident, un fișier ce anterior a fost deschis, iar apoi a fost închis poate fi din nou deschis cu oricare dintre fluxurile din programul în curs de elaborare, nu

neapărat doar cu cel cu care a fost asociat mai înainte. Cu alte cuvinte, fluxurile (fișierele logice) pot fi folosite pentru deschiderea oricăror fișiere externe, iar fișierele externe pot fi deschise de oricare dintre fluxurile din program.

Citirea datelor dintr-un fișier extern se efectuează cu ajutorul operatorului de extragere a datelor din fluxul de intrare asociat cu fișierul respectiv:

```
<Nume flux> >> <Variabilă> { >> <Variabilă> };
```

Exemple:

1) FI >> x;

2) FI >> x >> y;

3) FI >> x >> y >> z;

Scrierea datelor într-un fișier extern se efectuează cu ajutorul operatorului de includere a datelor în fluxul de ieșire asociat cu fișierul respectiv:

```
<Nume flux> << <Expresie> { << <Expresie> };
```

Exemple:

1) FE << x;

2) FE << "x=" << x;

3) FE << x " " << y << " " << z;

În concluzie, prezentăm ordinea în care trebuie apelate funcțiile destinate prelucrării datelor din fișiere:

1) **Asocierea** fluxului f cu fișierul extern s și deschiderea acestuia pentru citire/scriere: `ifstream f(s)`, respectiv, `ofstream f(s)`.

2) **Citirea/scrierea** unei componente a fișierului asociat cu fluxul f : $f >> v$, respectiv, $f << e$.

3) **Închiderea** fișierului asociat cu fluxul f : `f.close()`.

După închiderea fișierului, fluxul f poate fi asociat cu un alt fișier extern.

Întrebări și exerciții

- 1) Explicați termenii *fișier logic*, *fișier extern*.
- 2) Unde se păstrează datele unui fișier? Care este destinația asignării?
- 3) Cum se clasifică fișierele în funcție de operațiile permise și modul de acces?
- 4) Ce operații pot fi efectuate asupra fișierelor? Dar asupra componentelor unui fișier?
- 5) Pentru ce este necesară operația de asociere a unui fișier logic cu un fișier extern?
- 6) Pentru ce sunt necesare operațiile de deschidere și închidere a fișierelor? Cum se efectuează aceste operații?
- 7) Indicați ordinea în care trebuie efectuate operațiile de prelucrare a datelor dintr-un fișier de intrare? Dar dintr-un fișier de ieșire?

PASCAL

- 8 EXERSEAZĂ! Reprezentați pe un desen similar celui din *figura 1.8.* structura următoarelor tipuri de date:

a) **type** Tabel = **array** [1..5, 1..10] **of** real;
FisierTabele = **file of** Tabel;

b) **type** Multime = **set of** 'A'..'C';
FisierMultimi = **file of** Multime;

c) **type** Punct = **record** x, y:real **end**;
Segment = **record** A, B:Punct **end**;
FisierSegmente = **file of** Segment;

- 9 Explicați destinația procedurilor **read** și **write**. Ce tip trebuie să aibă variabila v într-un apel de forma **read** (f, v)? Ce tip trebuie să aibă expresia e într-un apel de forma **write**(f, e)?

- 10 ANALIZEAZĂ! Se consideră declarațiile:

```
type Numere = file of integer;  
var F, FIN, FOUT : Numere;  
    i, j, k : integer;  
    x, y, z : integer;
```

Explicați ce operații va efectua programul în procesul execuției următoarelor secvențe de instrucțiuni:

a) **assign**(F, 'EXP.DAT');
rewrite(F);
i:=10; **write**(F, i);
j:=20; **write**(F, j);
k:=30; **write**(F, k);
close(F);

b) **assign**(F, 'EXP.DAT');
reset(F);
read(F, x);
read(F, y);
read(F, z);
close(F);

c) **assign**(FOUT, 'EXP.DAT');
rewrite(FOUT);
i:=40; j:=50; k:=60;
write(FOUT, i); **write**(FOUT, j); **write**(FOUT, k);
close(FOUT);
assign(FIN, 'EXP.DAT');
reset(FIN);
read(FIN, x); **read**(FIN, y); **read**(FIN, z);
close(FIN);

- 11 În condițiile exercițiului precedent, ce valori vor lua variabilele x, y, z dacă secvențele de instrucțiuni a), b) și c) vor fi executate consecutiv, una după alta?

- 12 Variabilele A și B sunt introduse prin declarația

```
var A, B : file of integer;
```

Este oare corectă instrucțiunea?

```
A:=B
```

Argumentați răspunsul.

C++

- 13 Cum se declară fluxurile? Cum se asociază un flux cu un fișier extern?
14 ANALIZEAZĂ! Se consideră declarațiile:

```
int i, j, k,  
int x, y, z;
```

Explicați ce operații va efectua programul în procesul execuției următoarelor secvențe de instrucțiuni:

a)

```
ofstream F("EXP.DAT");  
i:=10; F << i << " ";  
j:=20; F << j << " ";  
k:=30; F << k;  
F.close();
```

b)

```
ifstream F("EXP.DAT");  
F >> x;  
F >> y;  
F >> z;  
F.close();
```

c)

```
ofstream FOUT("EXP.DAT");  
i:=40; j:=50; k:=60;  
FOUT << i << " " << j << " " << k;  
FOUT.close();  
ifstream FIN("EXP.DAT");  
FIN >> x >> y >> z;  
FIN.close();
```

- 15 ANALIZEAZĂ! În condițiile exercițiului precedent, ce valori vor lua variabilele x, y, z dacă secvențele de instrucțiuni a), b) și c) vor fi executate consecutiv, una după alta?
16 ANALIZEAZĂ! Variabilele A și B sunt introduse prin declarația

```
fstream A("text.in");  
ofstream B("test.out");
```

Este oare corectă instrucțiunea?

```
B=A
```

Argumentați răspunsul.

- ⑦ **DESCOPERĂ!** În afară de fluxurile de intrare (`ifstream`) și cele de ieșire (`ofstream`), limbajul C++ oferă utilizatorului și fluxuri de tipul `fstream`. Crearea unui flux de acest tip și asocierea lui cu un fișier extern se efectuează cu ajutorul următoarei construcții gramaticale:

```
fstream<Nume flux de intrare/ieșire> ("<Nume fișier extern>", <Opțiune>);
```

unde *<Opțiune>* specifică modul de deschidere a fluxului. Utilizând sistemul de asistență al mediului de dezvoltare a programelor și surse din Internet, aflați destinația opțiunilor ce pot fi utilizate la asocierea fluxurilor de acest tip cu fișierele externe. Completați tabelul de mai jos:

Opțiune	Destinația
<code>ios::in</code>	
<code>ios::out</code>	
<code>ios::app</code>	
<code>ios::ate</code>	
<code>ios::trunc</code>	
<code>ios::binary</code>	

- ⑧ **DESCOPERĂ!** În limbajul C++ există mai multe funcții destinate verificării stării curente a unui flux. Utilizând sistemul de asistență al mediului de dezvoltare a programelor și surse din Internet, aflați ce rezultate returnează aceste funcții și completați tabelul de mai jos:

Funcția	Destinația
<code>bad()</code>	
<code>fail()</code>	
<code>eof()</code>	
<code>good()</code>	
<code>clear()</code>	

1.8. Fișiere secvențiale

PASCAL

Fie definițiile PASCAL

```
type FT = file of T;
var f : FT; v : T;
```

prin care au fost introduse tipul *fișier* FT cu tipul de bază T, variabila de tip *fișier* *f* și variabila *v* de tipul T.

Pentru a deschide un **fișier secvențial de ieșire**, se apelează procedura `rewrite(f)`. În continuare în fișier se înscriu componentele respective. O componentă se înscrie printr-un apel de forma:

`write(f, e),`

unde e este o expresie de tipul T . O instrucțiune de forma

`write(f,  $e_1$ ,  $e_2$ , ...,  $e_n$ )`

este echivalentă cu secvența de instrucțiuni

`write(f,  $e_1$ ) ; write(f,  $e_2$ ); ...; write(f,  $e_n$ ).`

După înscrierea ultimei componente fișierul trebuie închis.

Exemplu:

```
Program P90;
{ Crearea unui fisier cu componente de tipul Elev }
type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
FisierElevi = file of Elev;
var FE : FisierElevi;
    E : Elev;
    str : string;
    i, n : integer;
begin
    write('Dati numele fisierului de creat: ');
    readln(str);

    assign(FE, str);      { asociaza FE cu numele din str }
    rewrite(FE);          { deschide FE pentru scriere }

    write('Dati numarul de elevi: '); readln(n);

    for i:=1 to n do
        begin
            writeln('Dati datele elevului ', i);

            { citește cimpurile variabilei E de la tastatura }
            write('Numele: ');      readln(E.Nume);
            write('Prenumele: ');  readln(E.Prenume);
            write('Nota medie: '); readln(E.NotaMedie);

            { scrie valoarea variabilei E in fisierul FE }
            write(FE, E);
        end;
    close(FE);              { inchide fisierul FE }
    readln;
end.
```

Pentru a **deschide un fișier secvențial de intrare**, se apelează procedura `reset(f)`. Componenta curentă se citește din fișier printr-un apel de forma:

```
read(f, v).
```

O instrucțiune de forma

```
read(f, v1, v2, ..., vn)
```

este echivalentă cu secvența de instrucțiuni

```
read(f, v1); read(f, v2); ..., read(f, vn).
```

Sfârșitul de fișier este semnalizat de funcția booleană `eof(f)`, care ia valoarea `true` după citirea ultimei componente.

Exemplu:

```
Program P90;
{ Citirea unui fișier cu componente de tipul Elev }
type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
    FisierElevi = file of Elev;
var FE : FisierElevi;
    E : Elev;
    str : string;
begin
    write('Dati numele fișierului de citit: ');
    readln(str);

    assign(FE, str);    { asociază FE cu numele din str }
    reset(FE);          { deschide fișierul FE pentru citire }

    while not eof(FE) do
        begin
            { citește E din fișierul FE }
            read(FE, E);
            { afișează E pe ecran }
            writeln(E.Nume, ' ', E.Prenume, ':',
                    E.NotaMedie : 5:2);
        end;
        close(FE);      { închide fișierul FE }
        readln;
    end.
```

Subliniem faptul că elaborarea programelor de prelucrare a fișierelor secvențiale este posibilă fără a cunoaște numărul de componente ale fișierelor respective. Teoretic, numărul de componente ce pot fi scrise într-un fișier secvențial de ieșire

este nelimitat. Practic însă, acest număr este limitat de capacitatea de memorare a suportului extern de informații. În cazul fișierelor de intrare, programul trebuie să termine citirea consecutivă a componentelor unui fișier secvențial imediat cum se ajunge la elementul *EOF*.

C++

În cazul unui fișier secvențial, componentele acestuia pot fi accesate numai una după alta, începând cu prima. Programul ce urmează ilustrează modul de creare a unui fișier secvențial.

```
// Crearea unui fișier secvențial
#include<iostream>    //pentru utilizare cout
#include<fstream>     //pentru utilizare ofstream
using namespace std;
int main()
{
    ofstream f („test.out”); // deschide fișier pentru scriere
    f << „Primul meu fișier!” << endl; //scrie in fișier
    cout << „Fișierul a fost creat” << endl;
    return 0;
}
```

După execuția acestui program, pe monitor va apare mesajul *Fișierul a fost creat*, iar în dosarul proiectului vom găsi fișierul *test.out* pe care îl putem citi cu orice editor de texte, de exemplu, cu aplicația *Notepad* din sistemul de operare *Windows*. De asemenea, fișierul nou-creat poate fi deschis și cu ajutorul comenzii *Open* din meniul *File* al mediului de dezvoltare a programelor C++. Evident, fișierul în cauză va conține textul *Primul meu fișier!*.

După cum s-a accentuat în paragraful precedent, în limbajul C++ fișierele sunt tratate ca secvențe de octeți. Prin urmare, gruparea datelor din fișiere în unități lexicale (numere întregi, numere reale, șiruri de caractere etc.) cade în sarcina programatorului. Pentru o înțelegere mai profundă a acestei abordări, în calitate de exemplu vom examina următoarea problemă practică, frecvent întâlnită în viața cotidiană.

Problemă: Pentru fiecare dintre elevii unei clase sunt cunoscute următoarele date: numele, prenumele, nota medie la o anumită disciplină școlară. Se cere ca datele respective să fie stocate într-un fișier secvențial și afișate pe ecran.

Rezolvare: În programul ce urmează datele respective sunt citite de la tastatură, se memorează în tabloul *e[50]* și, totodată, sunt stocate în fișierul *elevi.in*.

```
// Program P90
// Crearea fisierului ce contine date despre elevi
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
```

```

int main()
{
    ofstream f(„elevi.in”);
    struct elev
    {
        char nume [15];
        char prenume[20];
        float NotaMedie;
    };
    int i,n;
    elev e[50];
    cout<<"Dati numarul de elevi";
    cin>>n;
    cin.get();
    for (i=1;i<=n;i++)
    {
        cout<<"Introduceti numele elevului "<<i<<" ";
        cin.get(e[i].nume,15);
        cin.get();
        cout<<"Introduceti prenumele elevului "<<i<<" ";
        cin.get(e[i].prenume,20);
        cin.get();
        cout<<"Introduceti nota medie ";
        cin>>e[i].NotaMedie;
        cin.get();
        f<<e[i].nume<<' '<<e[i].prenume<<' '<<e[i].NotaMedie<<endl;
    }
    f.close();
    return 0;
}

```

În acest program, extragerea numelui și prenumelui elevului curent din fluxul de intrare `cin` sau, cu alte cuvinte, citirea valorilor variabilelor `Nume` și `Prenume` de la tastatură se efectuează cu ajutorul funcției `get`. Dacă lista de argumente din apelul acestei funcții nu este vidă, ea extrage (citește) din fluxul supus prelucrării un șir de caractere și îl depune în variabila respectivă. În cazul unui apel fără argumente, funcția `get` pur și simplu trece (mută cursorul) peste un caracter, în cazul nostru – peste caracterul special *EOL*. Amintim că caracterele *EOL* apar în fluxul `cin` la acționarea tastei *<ENTER>*.

Valoarea curentă a variabilei `NotaMedie` se extrage din fluxul de intrare `cin` cu ajutorul operatorului `>>`. Acest operator citește, începând cu poziția curentă a cursorului, un număr real din fluxul supus prelucrării și îl depune în variabila `NotaMedie`. După citire, cursorul va fi mutat după ultima cifră a numărului respectiv, în cazul nostru, în fața simbolului *EOL*. Pentru a muta cursorul peste acest simbol, se apelează din nou, evident, fără argumente, funcția `get`.

După citirea de la tastatură, datele despre elevul curent sunt incluse (scrise) în fluxul de ieșire `f`, care este asociat cu fișierul extern `elevi.in`.

Întrucât în limbajul C++ orice flux reprezintă doar o succesiune de caractere, fără a avea o anumită structură, simpla includere a șirurilor de caractere `Nume`, `Prenume` și a numărului real `NotaMedie` într-un flux de ieșire nu va fi suficientă. Problema constă în faptul că orice program care va citi fișierul respectiv nu va ști unde se termină șirul de caractere `Nume` și unde începe șirul de caractere `Prenume`. Pentru a rezolva această problemă, în programul de mai sus, în fluxul de ieșire, după fiecare din valorile `Nume`, `Prenume`, se include câte un spațiu, iar după fiecare din valorile `NotaMedie` – câte un caracter special *EOL* (sfârșit de linie).

Evident, după terminarea procesului de execuție a programului **P90**, datele din tabloul `e[5]` vor fi pierdute, pe când cele din fișierul nou-creat `elevi.in` vor fi păstrate pe suportul extern de informații. Citirea datelor din acest fișier și afișarea lor la ecran poate fi făcută cu ajutorul programului ce urmează.

```
// Program P91
// Citirea fisierului ce contine date despre elevi
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ifstream f („elevi.in”); //se deschide fisierul „elevi.in”
    struct elev
    {
        char nume [15];
        char prenume[20];
        float NotaMedie;
    };

    elev e;
    f>>e.nume>>e.prenume>>e.NotaMedie;
    while (!f.eof())
    {
        cout<<e.nume<<’ ,<<e.prenume<<’ ,<<e.NotaMedie<<endl;
        f>>e.nume>>e.prenume>>e.NotaMedie;
    }
    f.close(); //se inchide fisierul
    return 0;
}
```

Datele din fluxul de intrare `f` sunt extrase (citite) până când funcția `eof` nu va semnala faptul că s-a ajuns la caracterul *EOF*, adică la sfârșitul fișierului extern `elevi.in`.

Subliniem faptul că numărul de componente ale unui fișier nu este cunoscut din declarația tipului respectiv. Teoretic, într-un fișier secvențial de ieșire pot fi înscrise un număr infinit de componente. Practic, numărul componentelor este limitat de capacitatea de memorare a suportului extern de informație. Citirea consecutivă a componentelor unui fișier secvențial de intrare se încheie când se ajunge la elementul *EOF*.

Întrebări și exerciții

- 1 Câte componente poate avea un fișier? În ce ordine se scriu/citesc componentele unui fișier secvențial?
- 2 EXERSEAZĂ! Se consideră următoarele tipuri de date:

PASCAL

```
type Data = record
    Ziua : 1..31;
    Luna : 1..12;
    Anul : integer;
end;
Persoana = record
    NumePrenume : string;
    DataNasterii : Data;
end;
FisierPersoane = file of
    Persoana;
```

C++

```
struct Data
{
    int Ziua, Luna, Anul;
};
struct elev
{
    char NumePrenume [40];
    Data DataNasterii;
};
```

Elaborați un program care citește de la tastatură datele referitoare la n persoane și le înregistrează într-un fișier. Creați fișierele FILE1.PRS, FILE2.PRS, FILE3.PRS care trebuie să conțină datele referitoare, respectiv, la 2, 7 și 10 persoane.

- 3 REZOLVĂ! Elaborați un program care citește fișiere create de programul din exercițiul precedent și afișează pe ecran:
 - a) toate persoanele din fișier;
 - b) persoanele născute în anul a ;
 - c) persoanele născute pe data $z.l.a$;
 - d) persoana cea mai în vârstă;
 - e) persoana cea mai tânără.
- 4 REZOLVĂ! Elaborați un program care afișează pe ecran media aritmetică a numerelor înscrise într-un fișier de numere reale.
- 5 REZOLVĂ! Într-un fișier sunt înscrise caractere arbitrare. Elaborați un program care afișează pe ecran numărul vocalelor din fișier.
- 6 OBSERVĂ! Comentați următorul program:

PASCAL

```
Program P92;
{ Eroare }
type
    FisierNumere = file of
integer;
var FN : FisierNumere;
    i : integer;
    r : real;
    s : string;
begin
    Writeln('Dați numele
        fisierului: ');
    readln(s);
```

C++

```
// Program P92
// Eroare
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    char s[10];
    cout<<"Dați numele fisierului
        : \n";
    cin>>s;
    ifstream FN(s);
    int i;
```

```

assign(FN, s);
rewrite(FN);
i:=1;
write(FN, i);
i:=10;
write(FN, i);
r:=20;
write(FN, r);
close(FN);
end.

```

```

float r;
i=1;
FN<<i;
i=10;
FN<<i;
r=20;
FN<<r;
FN.close();
return 0;
}

```

1.9. Fișiere text

E cunoscut faptul că datele fișierelor logice din programele scrise într-un limbaj de nivel înalt se reprezintă prin secvențe de cifre binare. Acest mod de reprezentare a datelor este convenabil în cazul memoriilor externe (discurile și benzile magnetice, discurile optice, memoriile de tip *flash* ș.a.). În cazul echipamentelor de intrare/ieșire (tastatura, ecranul, imprimanta etc.), datele respective trebuie reprezentate într-o formă înțeleasă de om, și anume, prin secvențe de caractere.

Pentru a facilita interacțiunea dintre om și sistemul de calcul, în limbajele de nivel înalt informația destinată utilizatorului se reprezintă în formă de fișiere *text*. Un fișier *text* este format dintr-o secvență de caractere divizată în linii (*fig. 1.9*). Lungimea liniilor este variabilă. Sfârșitul fiecărei linii este indicat de un element special, notat *EOL* (*End Of Line* – sfârșit de linie). Întrucât lungimea liniilor este variabilă, poziția unei linii în cadrul oricărui fișier *text* nu poate fi calculată din timp. În consecință, accesul la componentele fișierelor *text* este **secvențial**. Cu alte cuvinte, scrierea sau citirea unei componente, adică a unui caracter sau a unei linii, este posibilă doar după ce a fost citită sau scrisă componenta precedentă.

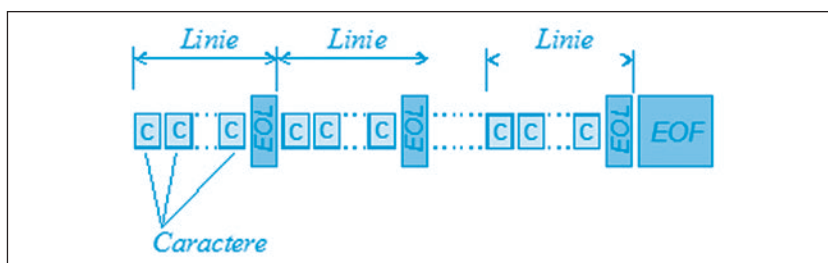


Fig. 1.9. Structura unui fișier text

În cadrul disciplinei școlare Informatica se studiază următoarele modalități de scriere/citire a fișierelor *text*:

- la nivel de caractere;
- la nivel de lexeme;
- la nivel de linii.

În cazul **prelucrărilor la nivel de caractere**, datele din fișierul extern sunt citite/scrise caracter după caracter, fiecare dintre caractere fiind păstrat într-o variabilă de tip `char`. Asamblarea acestor caractere în numere sau șiruri de caractere cade în sarcina programatorului.

De asemenea, tot programatorul trebuie să aibă grijă de prelucrarea elementelor speciale *EOL* (sfârșit de linie).

În cazul **prelucrărilor la nivel de lexeme**, scrierea și citirea datelor se efectuează cu ajutorul unor funcții și proceduri care „asamblează” datele din fișierul extern în numere și șiruri și le păstrează în variabile întregi, reale, caracteriale sau șiruri de caractere, după caz. Evident, lexemele în cauză trebuie să fie delimitate prin caractere albe (spațiu, tabulator sau sfârșit de linie).

De exemplu, presupunem că fișierul de intrare conține următoarele date:

```
82 3.14 ACCEPTAT<EOL><EOF>
```

Intuitiv, este clar că fișierul de intrare conține numărul întreg 82, numărul real 3,14 și șirul de caractere *ACCEPTAT*.

Prelucrarea acestui fișier la nivel de caractere constă în citirea consecutivă a fiecăruia dintre caracterele ce se conțin în el:

```
8 2   3 . 1 4   A C C E P T A T
```

și depunerea lor în variabilele de tip `char` din program. În continuare, programatorul va trebui să scrie suplimentar o secvență de instrucțiuni care transformă caracterele citite în valorile de care el are nevoie.

Prelucrarea aceluiași fișier la nivel de lexeme constă în citirea caracterelor din fișierul de intrare și gruparea acestora în trei lexeme:

```
82 3.14 ACCEPTAT
```

Ulterior, lexemele astfel obținute sunt în mod automat transformate în reprezentarea internă și depuse în variabilele de tipul respectiv: *integer/int*, *real/float* și *șir de caractere*.

În cazul **prelucrărilor la nivel de linii**, datele din fișierul extern sunt citite/scrise linie după linie, fiecare din linii fiind păstrate într-o variabilă de tip *șir de caractere*.

În general, prelucrările la nivel de caractere sunt utile în programarea algoritmilor de procesare a textelor, de exemplu, secvențierea în cuvinte și propoziții, împărțirea cuvintelor în silabe etc., iar cele de prelucrare la nivel de lexeme – în cazul programării algoritmilor de procesare a datelor preponderent numerice. Prelucrările la nivel de linii oferă utilizatorului posibilitatea să manipuleze atât cu caractere separate, cât și cu lexeme, utilizând în acest scop funcțiile predefinite, destinate procesării șirurilor de caractere.

PASCAL

În limbajul PASCAL, un fișier *text* se definește printr-o declarație de forma:

```
var f : text;
```

în care tipul predefinit `text` fiind cunoscut oricărui program. Subliniem faptul că tipurile `text` și `file of char` sunt distincte, întrucât fișierul `file of char`

nu include elemente *EOL*. Pentru a ne convinge de acest lucru este suficient să comparăm structurile fișierelor respective, prezentate în *figurile 1.8 și 1.9*.

Fișierele de tip *text* pot fi prelucrate cu ajutorul procedurilor cunoscute, aplicabile oricărui tip de fișiere: *assign*, *reset*, *rewrite*, *read*, *write*, *close*. În completare, limbajul PASCAL include proceduri speciale, destinate prelucrării elementelor *EOL*:

writeln(f) – înscrie în fișier elementul *EOL* (sfârșit de linie);

readln(f) – trece la linia următoare.

Sfârșitul de linie este semnalat de funcția booleană *eoIn(f)*, care ia valoarea *true* după citirea ultimului caracter din linie.

O instrucțiune de forma

```
writeln(f, e1, e2, ..., en)
```

este echivalentă cu secvența de instrucțiuni

```
write(f, e1, e2, ..., en); writeln(f).
```

O instrucțiune de forma

```
readln(f, v1, v2, ..., vn)
```

este echivalentă cu secvența de instrucțiuni

```
read(f, v1, v2, ..., vn); readln(f).
```

Pentru introducerea și extragerea datelor, de regulă, se utilizează fișierele *text* predefinite *Input* și *Output*, cunoscute oricărui program PASCAL. Fișierul *Input* este destinat numai pentru operații de citire și este asociat cu fișierul de intrare al sistemului de operare (de regulă, tastatura). Fișierul *Output* este destinat numai pentru operații de scriere și este asociat cu fișierul standard de ieșire al sistemului de operare (de regulă, ecranul). Aceste fișiere sunt deschise și închise automat la începutul și, respectiv, la sfârșitul execuției programului. Dacă numele fișierului nu este specificat în lista parametrilor unui apel de subalgoritm, se presupune că fișierul *text* implicit este *Input* sau *Output*, în funcție de natura subalgoritmului. De exemplu, *read(c)* este echivalent cu *read(Input, c)*, iar *write(c)* este echivalent cu *write(Output, c)*.

Pentru exemplificare, prezentăm în continuare un program care creează fișierul extern *LISTA.TXT* ce conține numele și prenumele a *n* elevi, numărul acestora nefiind cunoscut la momentul scrierii programului.

Program P93;

```
{ Crearea fisierului text LISTA.TXT prin }  
{ prelucrari la nivel de linii }  
var F : text;           { fisierul logic }  
    n : integer;        { numarul de elevi }  
    NP : string;        { numele si prenumele elevului }  
    i : integer;        { contor ciclu }
```

```

begin
  { asocierea fisierului logic F cu fisierul extern }
  assign(F, 'LISTA.TXT');
  { Deschiderea fisierului F pentru scriere }
  rewrite(F);
  { Citirea numarului de elevi de la tastatura }
  write('Dati numarul de elevi n=');
  readln(n);
  { Scrierea numarului de elevi in fisierul F }
  writeln(F, n);
  { ciclu pe numarul de elevi }
  for i:=1 to n do
    begin
      { Citirea numelui si prenumelui de la tastatura }
      writeln('Dati numele si prenumele elevului ', i, ':');
      readln(NP);
      {Scrierea numelui si prenumelui in fisierul F }
      writeln(F, NP);
    end;
  { Inchiderea fisierului F }
  close(F);
  writeln('Ati introdus ', n, ' elevi');
  readln;
end.

```

În continuarea exemplului precedent, prezentăm mai jos un program care citește conținutul fișierului extern LISTA.TXT și îl afișează pe ecran.

```

Program P94;
{ Citirea fisierului text LISTA.TXT prin }
{ prelucrari la nivel de linii }
var F : text;           { fisierul logic F }
    n : integer;        { numarul de elevi }
    NP : string;        { numele si prenumele }
    i : integer;        { contor ciclu }

begin
  { Asocierea fisierului logic F cu fisierul extern }
  assign(F, 'LISTA.TXT');
  { Deschiderea fisierului F pentru citire }
  reset(F);
  { Citirea numarului de elevi n din fisierul F }
  readln(F, n);
  { Afisarea numarului de elevi n la ecran }
  writeln('Numarul de elevi n=', n);
  { Ciclu pe numarul de elevi }
  for i:=1 to n do

```

```

begin
  readln(F, NP); { citirea NP din fisierul F }
  writeln(NP);   { afisarea NP la ecran      }
end;
close(F);        { inchiderea fisierului F  }
readln;
end.

```

Prelucrări la nivel de caractere

Fișierele *text* pot fi citite și scrise caracter cu caracter. În acest scop, variabila *v* din apelurile *read(f, v)* și *write(f, v)* trebuie să fie de tip *char*. Evident, prelucrarea caracterelor speciale *EOL* și *EOF* cade în sarcina programatorului.

Pentru exemplificare, prezentăm programul P95, care creează pe discul curent fișierul *text* *FILE.TXT*. și depune în el liniile citite de la tastatură. La momentul scrierii programului, lungimea fiecăreia dintre linii și numărul acestora nu se cunoaște. La introducerea datelor de la tastatură, sfârșitul liniei curente se indică prin acționarea tastei *<ENTER>*, iar sfârșitul fișierului prin acționarea tastelor *<CTRL+Z>*, *<ENTER>*.

```

Program P95;
{ Crearea fisierului text FILE.TXT prin prelucrari      }
{ la nivel de caractere                                 }
var F : text;          { fisierul logic F              }
    C : char;
begin
  assign(F, 'FILE.TXT'); { asociaza F cu FILE.TXT      }
  rewrite(F);            { deschide F pentru scriere    }
  while not eof do       { verifica daca-i <CTRL+Z>     }
  begin
    while not eoln do    { verifica daca-i <ENTER>      }
    begin
      read(C);           { citeste C de la tastatura   }
      write(F, C);       { scrie C in fisierul F        }
    end;
    writeln(F);          { scrie EOL in fisierul F      }
    readln;              { trece la linia urmatoare    }
  end;
  close(F);              { scrie EOF in fisierul F      }
end.

```

Accentuăm faptul că în programul de mai sus componentele fișierului de intrare *Input* (tastatura) se citesc la nivel de caractere, iar sfârșitul fiecăreia dintre linii (elementul special *EOL*) se depistează cu ajutorul funcției *eoln*. Într-un mod similar, sfârșitul fișierului de intrare (elementul special *EOF*) se depistează cu ajutorul funcției *eof*.

Programul ce urmează afișează conținutul fișierului extern *FILE.TXT* pe ecran.

```

Program P96;
{ Citirea fisierului text FILE.TXT prin prelucrari }
{ la nivel de caractere }
var F : text;           { fisierul logic F }
    C : char;
begin
    assign(F, 'FILE.TXT'); { asociaza F cu FILE.TXT }
    reset(F);              { deschide F pentru citire }
    while not eof(F) do    { testeaza daca-i EOF in F }
        begin
            while not eoln(F) do { testeaza daca-i EOL in F }
                begin;
                    read(F, C);    { citeste C din F }
                    write(C);      { afiseaza C la ecran }
                end;
                readln(F);         { in F trece la linia urmatoare }
                writeln;           { pe ecran trece la o linie noua }
            end;
        close(F);                { inchide fisierul F }
        readln;
    end.

```

În programul de mai sus, componentele fișierului de ieșire **Output** (ecranul) se scriu la nivel de caractere. Sfârșitul *EOL* al fiecăreia dintre linii se înscrie prin apelul procedurii `readln`, iar sfârșitul de fișier *EOF* – prin apelul procedurii `close`.

Prelucrări la nivel de lexeme

Explorarea caracter cu caracter a fișierelor *text* este greoaie în situația în care secvențele de caractere din text trebuie interpretate ca formând date de tip **integer**, **real**, **boolean**, *șir de caractere*. Conversia între forma externă și reprezentarea internă a acestor tipuri ar cădea în sarcina programatorului. Pentru a ușura munca acestuia, procedurile de citire/înscriere au fost extinse în felul următor.

În cazul fișierelor *text*, variabila v dintr-un apel `read(f, v)` poate fi de tipul **integer**, **real**, **char** sau *șir de caractere*. La citire, secvența de caractere care reprezintă variabila v va fi transformată în reprezentarea internă.

Expresia e dintr-un apel `write(f, e)` poate fi urmată de specificatori de format. Valoarea expresiei poate fi de tipul **integer**, **real**, **boolean**, **char** sau *șir de caractere*. La scriere, valoarea respectivă este transformată din reprezentarea internă într-o secvență de caractere.

Secvențele de caractere citite/scrise de procedurile `read/write` se conformează sintaxei constantelor de tipul variabilei/expresiei v/e .

Pentru exemplificare, prezentăm programul P97 care:

- 1) citește de la tastatură o linie ce conține numere reale a, b, c – lungimile laturilor unui triunghi;
- 2) scrie numerele citite în linia curentă a fișierului *text* IN.TXT;
- 3) repetă pașii 1-2 până când utilizatorul nu va tasta <CTRL+Z> <ENTER>;
- 4) citește din linia curentă a fișierului *text* IN.TXT numerele reale a, b, c ;
- 5) calculează semiperimetrul p și aria s ale triunghiului respectiv;

- 6) scrie numerele reale a, b, c, p, s în fișierul *text* OUT.TXT;
- 7) repetă pașii 4-6 până când va fi citită ultima linie din fișierul *text* OUT.TXT;
- 8) afișează conținutul fișierului OUT.TXT pe ecran.

Program P97;

```
{ Prelucrarea triunghiurilor }
{ Scrierea si citirea la nivel de lexeme }
var F, G : text;           { fisiere logice }
    a, b, c, p, s : real;
    Str : string;
begin

    { Citirea datelor de la tastatura si stocarea lor in F }
    assign(F, 'IN.TXT'); { asociaza F cu IN.TXT }
    rewrite(F);          { deschide F pentru scriere }
    writeln('Dati numerele reale a, b, c:');
    while not eof do
        begin
            readln(a, b, c); { citeste a, b, c de la tastatura }
            writeln(F, a:8:2, b:8:2, c:8:2); { scrie a, b, c in F }
        end;
    close(F);              { inchide F }

    { Citirea datelor din F, efectuarea calculelor si }
    { stocarea rezultatelor in G }
    reset(F);              { deschide F pentru citire }
    assign(G, 'OUT.TXT'); { asociaza G cu OUT.TXT }
    rewrite(G);            { deschide G pentru scriere }
    while not eof(F) do
        begin
            readln(F, a, b, c); { citeste a, b, c din F }
            write(G, a:8:2, b:8:2, c:8:2); { scrie a, b, c in G }
            p:=(a+b+c)/2;
            s:=sqrt(p*(p-a)*(p-b)*(p-c));
            writeln(G, p:15:2, s:15:4); { scrie p, s în G }
        end;
    close(F);              { inchide F }
    close(G);              { inchide G }

    { Citirea rezultatelor din G si afisarea lor la ecran }
    reset(G);              { deschide G pentru citire }
    while not eof(G) do
        begin
            readln(G, Str); { citeste Str din G }
            writeln(Str);    { afisează Str pe ecran }
        end;
    close(G);              { inchide G }
    readln;
end.
```

Pentru datele de intrare

```
1 1 1 <ENTER>
3 4 6 <ENTER>
<CTRL+Z ><ENTER>
```

programul P97 afișează pe ecran:

```
1.00 1.00 1.00 1.50 0.4330
3.00 4.00 6.00 6.50 5.3327
```

Accentuăm faptul că programul de mai sus este destinat pentru prelucrarea datelor referitoare la un număr necunoscut de triunghiuri, fapt ce-i oferă o flexibilitate mai mare în utilizare. Această flexibilitate se datorează utilizării funcției `eof`. Cu ajutorul acestei funcții se testează dacă în procesul citirii s-a ajuns la sfârșitul fiecăruia dintre fișierele `Input` (tastatura) și `IN.TXT`.

C++

În limbajul C++, un fișier text conține numai caractere ASCII organizate pe linii de lungimi diferite. Elementul *EOL* se reprezintă prin caracterul *escape* “\n” (*new line*, linie nouă). Sfârșitul fișierului este specificat prin elementul *EOF*. Întrucât lungimea liniilor este variabilă, poziția unei linii în cadrul fișierului nu poate fi calculată din timp. În consecință, accesul la componentele fișierelor *text* este secvențial.

Întrucât limbajul C++ nu impune nicio structură asupra conținutului fișierelor, secvențierea lor în lexeme (numere, șiruri de caractere) și linii cade în sarcina programatorului. În acest scop pot fi folosite toate funcțiile destinate prelucrării fișierelor secvențiale studiate în paragraful 1.8.

Pentru exemplificare, prezentăm în continuare programul P93, care creează pe discul curent un fișier *text*, numele căruia este dat de la tastatură. Liniile fișierului sunt delimitate prin acționarea tastei `<ENTER>`, iar sfârșitul fișierului – prin acționarea tastelor `<CTRL+Z ><ENTER>`.

```
// Program P93
// Crearea unui fișier text
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    char nume_fisier[80];
    char C;
    cout << "\n\tProgramul creeaza un fisier text\n";
    cout << "\n\tIntroduceti numele fisierului: ";
    cin.getline(nume_fisier, 80);
    ofstream f(nume_fisier);
    cout << "\n\tIntroduceti caractere. Sfarsit Enter CTRL+Z\n";
    while(cin.get(c))
        f << c;
    f.close();
    return 0;
}
```

Prelucrări la nivel de caractere

Reamintim că la citirea cu operatorul >> sunt ignorate caracterele albe (spațiul, tabulatorul, sfârșitul de linie). Prin urmare, dacă intenționăm să citim unul după altul toate caracterele din fișier, inclusiv cele albe, vom utiliza funcția `get()`.

```
// Program P94
// Citirea dintr-un fișier text
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    char nume_fisier[80];
    char c ;
    cout << "\n\tProgramul citește un fișier text\n";
    cout << "\n\tIntroduceți numele fișierului: ";
    cin.getline(nume_fisier,80);
    ifstream f(nume_fisier);
    f.get(c);
    while(!f.eof())
    {
        f.get(c);
        cout<<c;
    }
    f.close();
    return 0;
}
```

Pentru a verifica dacă poziția curentă de citire/scriere a ajuns la sfârșitul unui fișier, se folosește funcția `eof()`, care întoarce valoarea 0 dacă poziția curentă nu este la sfârșitul fișierului și o valoare diferită de 0 dacă poziția actuală indică sfârșitul de fișier.

Programul de mai jos copiază caracter cu caracter datele din fișierul *text* IN.TXT în fișierul *text* OUT.TXT.

```
// Program P95
// Copierea caracter cu caracter a fișierelor text
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    char c ;
    ifstream f("in.txt"); // fluxul f se asociază cu in.txt
    ofstream g("out.txt"); // fluxul g se asociază cu out.txt
    f.get(c); // din f se citește primul caracter
    while(!f.eof()) // ciclu pe caracterele din f
    {
```

```

        g.put(c);           // caracterul citit se scrie in g
        f.get(c);          // din f se citeste caracterul urmator
    }
    f.close();              // inchiderea fisierului in.txt
    g.close();              // inchiderea fisierului out.txt
    return 0;
}

```

Prelucrări la nivel de linii

În cazul în care şirurile de caractere din fişierul de intrare, inclusiv caracterele albe, trebuie citite linie cu linie, se utilizează funcţia `getline`.

Programul de mai jos copiază linie cu linie datele din fişierul *text* IN.TXT în fişierul *text* OUT.TXT.

```

// Program P96
// Copierea linie cu linie a fisierelor text
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    char linie[80];
    ifstream f("in.txt");      // fluxul f se asociaza cu in.txt
    ofstream g("out.txt");      // fluxul g se asociaza cu out.txt
    while(!f.eof())            // ciclu pe liniile din f
    {
        f.getline(linie,80);    // din f se citeste linia curenta
        g<<linie<<endl;        // linia citita este scrisa in g
    }
    f.close();                 // inchiderea fisierului in.txt
    g.close();                 // inchiderea fisierului out.txt

    return 0;
}

```

Prelucrări la nivel de lexeme

Programul ce urmează utilizează funcţiile destinate prelucrării fişierelor C++ pentru secvenţierea acestora în lexeme. Mai exact, acest program:

- 1) citeşte de la tastatură o linie ce conţine numere reale a , b , c – lungimile laturilor unui triunghi;
- 2) scrie numerele citite în linia curentă a fişierului *text* in.txt;
- 3) repetă paşii 1-2 până când utilizatorul nu va tasta `<CTRL+Z>` `<ENTER>`;
- 4) citeşte din linia curentă a fişierului *text* in.txt numerele reale a , b , c ;
- 5) calculează semiperimetrul p şi aria s ale triunghiului respectiv;
- 6) scrie numerele reale a , b , c , p , s în fişierul *text* out.txt;

- 7) repetă pașii 4-6 până când nu va fi citită ultima linie din fișierul *text out.txt*;
8) afișează conținutul fișierului *out.txt* pe ecran.

```
//Program P97
//Prelucrarea triunghiurilor
//Scrierea si citirea la nivel de lexeme
#include <iostream>
#include <iomanip>
#include <cmath>
#include <fstream>
using namespace std;
int main()
{
    float a,b,c;
    float p,s;
    int i,n;
    char linie[80];

    //Citirea datelor de la tastatura si stocarea lor in f
    ofstream f("in.txt"); //deschide f pentru scriere
    cout<<"Dati numerele reale a,b,c \n";
    cin>>a>>b>>c;          // citeste a, b, c de la tastatura
    while (!cin.eof())      //repete pana la tastarea <CTRL+Z >
        <ENTER>
    {
        f<<a<<" "<<b<<" "<<c<<endl; // scrie a, b, c in f
        cin>>a>>b>>c;          // citeste a, b, c de la tastatura
    }
    f.close();              // inchide f

    //Citirea datelor din f, efectuarea calculelor si
    // stocarea rezultatelor in g
    ifstream f("in.txt");   // deschide f pentru citire
    ofstream g("out.txt");  // deschide g pentru scriere
    f>>a>>b>>c;              // citeste a,b,c din f
    while(!f.eof())
    {
        g<<a<<" "<<b<<" "<<c<<" "; // scrie a,b,c in g
        p=(a+b+c)/2;
        s=sqrt(p*(p-a)*(p-b)*(p-c));
        g<<p<<" "<<s<<"\n"; // scrie p si s in g
        f>>a>>b>>c;          // citeste a,b,c din f
    }
    f.close();              // inchide f
    g.close();              // inchide g
}
```

```
// Citirea rezultatelor din g si afisarea lor la ecran
ifstream g("out.txt"); // deschide g pentru citire
while(!g.eof())
{
    g.getline(linie,80); // citeste din g linia curenta
    cout<<linie<<endl; // afiseaza linia citita la ecran
}
g.close(); // inchide g
return 0;
}
```

Pentru datele de intrare

```
1 1 1 <ENTER>
3 4 6 <ENTER>
<CTRL+Z ><ENTER>
```

programul P97 afișează pe ecran:

```
1 1 1 1.5 0.433013
3 4 6 6.5 5.332684
```

Accentuăm faptul că programul de mai sus este destinat pentru prelucrarea datelor referitoare la un număr necunoscut de triunghiuri, fapt ce-i oferă o flexibilitate mai mare în utilizare. Această flexibilitate se datorează utilizării funcției `eof`. Cu ajutorul acestei funcții se testează dacă în procesul citirii s-a ajuns la sfârșitul fiecăruia dintre fișierele `Input` (tastatura) și `IN.TXT`.

Întrebări și exerciții

- ❶ EXERSEAZĂ! Desenați o schiță care redă structura unui fișier *text*. Explicați destinația fiecărui element grafic al schiței respective.
- ❷ ANALIZEAZĂ! Se consideră următorul fișier *text* de intrare:

```
Date 9.8 12 23.67 STOP<EOL><EOF>
```

Explicați cum vor fi citite datele din acest fișier și ce transformări vor avea loc în cazul următoarelor prelucrări: (a) la nivel de caractere; (b) la nivel de lexeme.

- ❸ ANALIZEAZĂ! Se consideră următorul fișier *text* de intrare:

```
5.1          9.3          Admis<EOL>6.4          4.3
Respins<EOL><EOF>
```

Explicați cum vor fi citite datele din acest fișier și ce transformări vor avea loc în cazul următoarelor prelucrări: (a) la nivel de caractere; (b) la nivel de lexeme.

- ❹ CREEAZĂ! Dați exemple de fișiere de intrare ce conțin datele inițiale ale unei probleme din fizică și ale alteia din biologie și descrieți procesul de citire a datelor din ele la nivel de caractere și la nivel de lexeme.

- ⑤ ANALIZEAZĂ! Care este diferența dintre prelucrarea unui fișier *text* la nivel de caractere și prelucrarea aceluiași fișier la nivel de lexeme? Dați exemple de algoritmi ce necesită implementarea prin prelucrări la nivel de caractere și prelucrări la nivel de lexeme.
- ⑥ ÎNVAȚĂ SĂ ÎNVEȚI! Pornind de la cele două modalități de citire a fișierelor de intrare la nivel de caractere și la nivel de lexeme, explicați modalitățile respective în cazul fișierelor *text* de ieșire. Dați exemple.
- ⑦ EXPLICĂ! Explicați semnificația elementelor *EOL* și *EOF*. Cum sunt prelucrate aceste elemente?
- ⑧ EXPLICĂ! Explicați cum se efectuează scrierea și citirea datelor în cazul fișierelor *text*. Care proceduri/funcții sunt utilizate în acest scop?
- ⑨ OBSERVĂ! Lansați în execuție următorul program:

PASCAL

```

Program P98;
{ Asocierea fișierului FN cu
  consola }
type FisierNumere = file of in-
teger;
var FN : FisierNumere;
    i : integer;
begin
  assign(FN, 'CON');
  rewrite(FN);
  i:=1;
  write(FN, i);
  i:=2;
  write(FN, i);
  i:=3;
  write(FN, i);
  close(FN);
  readln;
end.

```

C++

```

// Program P98
// Asocierea fișierului FN
// cu consola
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
  int i;
  ofstream FN („CON");
  i=1;
  FN<<i;
  i=2;
  FN<<i;
  i=3;
  FN<<i;
  FN.close();
  return 0;
}

```

Explicați rezultatele afișate pe ecran.

- ⑩ ANALIZEAZĂ! Se consideră fișierul logic *f*. Care din următoarele instrucțiuni destinate depistării sfârșitului de fișier sunt corecte?

PASCAL

- a) **if** eoln(f) **then**
 write ('Sfarsit de fisier')
 else **write** ('Nu-i sfarsit de fisier');
- b) **if** eof(f) **then**
 write ('Sfarsit de fisier')
 else **write** ('Nu-i sfarsit de fisier');
- c) **if** eoln(f)=false **then**
 write ('Sfarsit de fisier')
 else **write** ('Nu-i sfarsit de fisier');
- d) **if not** eof(f) **then**
 write ('Sfarsit de fisier')
 else **write** ('Nu-i sfarsit de fisier');

C++

- | | |
|----|--|
| a) | <pre>if (eof(f)) cout << "Sfarsit de fisier"; else cout << "Nu-i sfarsit de fisier";</pre> |
| b) | <pre>if (f.eof()) cout << "Sfarsit de fisier"; else cout << "Nu-i sfarsit de fisier";</pre> |
| c) | <pre>if (!eof(f)) cout << "Sfarsit de fisier"; else cout << "Nu-i sfarsit de fisier";</pre> |
| d) | <pre>if (!f.eof()) cout << "Nu-i sfarsit de fisier"; else cout << "Sfarsit de fisier";</pre> |

- ⑪ **APLICĂ!** Elaborați un program care afișează pe ecran conținutul oricărui fișier *text*.
- ⑫ **REZOLVĂ!** Elaborați un program care afișează pe ecran numărul de vocale dintr-un fișier *text*.
- ⑬ **PROGRAMEAZĂ!** Datele de intrare ale unui program sunt înmagazinate într-un fișier *text*. Fiecare linie a fișierului conține două numere întregi și trei numere reale separate prin spații. Elaborați un program care afișează suma numerelor întregi și suma numerelor reale din fiecare linie pe ecran.
- ⑭ **PROGRAMEAZĂ!** Datele de intrare ale unui program sunt înmagazinate într-un fișier *text*. Fiecare linie a fișierului conține trei numere reale separate prin spațiu și unul din cuvintele ADMIS, RESPINS. Elaborați un program care:
 - a) afișează conținutul fișierului în studiu pe ecran;
 - b) creează o copie de rezervă a fișierului;
 - c) creează un fișier *text* liniile căruia conțin media celor trei numere reale din liniile respective ale fișierului de intrare;
 - d) afișează pe ecran liniile fișierului de intrare, precedate de numerele de ordine 1, 2, 3 ș.a.m.d.
- ⑮ **PROGRAMEAZĂ!** Fiecare linie a unui fișier *text* conține următoarele date, separate prin spații:
 - numărul de ordine (număr întreg);
 - numele (un șir de caractere ce nu conține spații);
 - prenumele (un șir de caractere ce nu conține spații);
 - nota la disciplina 1 (număr real);
 - nota la disciplina 2 (număr real);
 - nota la disciplina 3 (număr real).

Elaborați un program care:

 - a) creează o copie de rezervă a fișierului în studiu;
 - b) afișează conținutul fișierului pe ecran;
 - c) creează un fișier *text* liniile căruia conțin următoarele date separate prin spații:
 - numărul de ordine (număr întreg);
 - numele (un șir de caractere ce nu conține spații);
 - prenumele (un șir de caractere ce nu conține spații);
 - nota medie (număr real);

Fișierul creat în punctul c trebuie afișat pe ecran.

2.1. Cantitatea de informație

Sensul uzual al cuvântului **informație** „știre, comunicare verbală, scrisă sau transmisă prin alte metode despre anumite fapte, evenimente, activități etc.” se concretizează într-un compartiment special al matematicii, denumit **teoria informației**. Conform acestei teorii, **sursa de informație** se descrie printr-o variabilă S care poate lua valori dintr-o mulțime finită de elemente distincte $\{s_1, s_2, \dots, s_n\}$. Se consideră că valorile curente ale variabilei S nu sunt cunoscute din timp. E cunoscută numai mulțimea $\{s_1, s_2, \dots, s_n\}$, **denumită mulțimea mesajelor posibile**.

De exemplu, semaforul de circulație poate fi reprezentat ca o sursă de informație, mulțimea de mesaje posibile ale căruia este $\{\text{verde, galben, roșu}\}$. Aparatul de telegrafiat reprezintă o sursă de informație, mulțimea de mesaje posibile ale căruia include literele $A, B, C, \dots, Z$, cifrele $0, 1, 2, \dots, 9$ și semnele de punctuație. Mesajele posibile ale tastaturii sunt: „Este acționată tasta A”, „Este acționată tasta B”, ..., „Este acționată tasta F1”, „Este acționată tasta F2”, ..., „Sunt acționate concomitent tastele CTRL și BREAK” etc.

Mesajele se transmit de la sursă către destinatar printr-un mediu fizic, numit **canal de transmisie** (fig. 2.1). De exemplu, mesajele telegrafice se transmit prin fir, mesajele radio prin eter, mesajele tastaturii printr-un set de conductori. **Perturbațiile** (zgomotele) din mediul fizic amintit pot altera mesajele transmise.

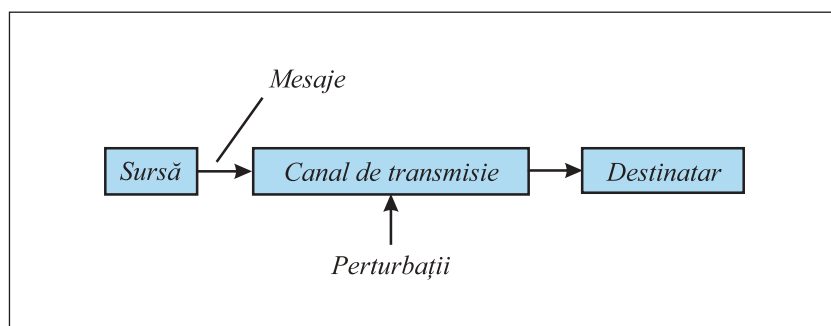


Fig. 2.1. Schema generală a unui sistem de transmisie a informației

Evident, valoarea curentă a variabilei S devine cunoscută destinatarului numai după recepționarea mesajului respectiv.

Cantitatea de informație I ce este conținută într-un mesaj emis de sursă se determină din relația:

$$I = \log_a n,$$

unde n este numărul de mesaje posibile ale sursei. Valoarea concretă a constantei a se stabilește prin alegerea **unității de măsură a cantității de informație**. De obicei, ca unitate de măsură se utilizează **bitul**.

Un bit este cantitatea de informație din mesajul unei surse cu numai două mesaje posibile.

Prin urmare, ca și în cazul altor mărimi (lungimea, masa, temperatura etc.), cantitatea de informație se măsoară prin compararea cu **etalonul**. Întrucât pentru sursa-etalon $n = 2$, din ecuația:

$$\log_a 2 = 1 \text{ (bit)}$$

obținem $a = 2$. În consecință, cantitatea de informație I , măsurată în biți, se determină din relația:

$$I = \log_2 n \text{ (bit)}.$$

În *tabelul 2.1* sunt prezentate valorile frecvent utilizate ale funcției $\log_2 n$.

Tabelul 2.1

Valorile funcției $\log_2 n$

n	$\log_2 n$	n	$\log_2 n$
1	0,000	21	4,392
2	1,000	22	4,459
3	1,585	23	4,524
4	2,000	24	4,585
5	2,322	25	4,644
6	2,585	26	4,700
7	2,807	27	4,755
8	3,000	28	4,807
9	3,170	29	4,858
10	3,322	30	4,907
11	3,459	31	4,954
12	3,585	32	5,000
13	3,700	33	5,044
14	3,807	34	5,087
15	3,907	35	5,129
16	4,000	36	5,170
17	4,087	37	5,209
18	4,170	38	5,248
19	4,248	39	5,285
20	4,322	40	5,322

Să analizăm câteva exemple. Cantitatea de informație a unui mesaj de semafor este de

$$I = \log_2 3 \approx 1,585 \text{ biți.}$$

Cantitatea de informație a unei litere a alfabetului latin $\{A, B, C, \dots, Z\}$, $n = 26$, este de

$$I = \log_2 26 \approx 4,700 \text{ biți.}$$

Cantitatea de informație a unei litere a alfabetului grec $\{A, B, \Gamma, \Delta, \dots, \Omega\}$, $n = 24$, este de

$$I = \log_2 24 \approx 4,585 \text{ biți.}$$

Dacă se cunoaște cantitatea de informație I ce este conținută într-un mesaj, cantitatea totală de **informație emisă** de sursă se determină din relația:

$$V = N I,$$

unde N este numărul de mesaje transmise.

Cantitățile mari de informație se exprimă prin multipli unui bit:

$$1 \text{ Kilobit (Kbit)} = 2^{10} = 1\,024 \text{ biți } (\approx 10^3 \text{ biți});$$

$$1 \text{ Megabit (Mbit)} = 2^{20} = 1\,048\,576 \text{ biți } (\approx 10^6 \text{ biți});$$

$$1 \text{ Gigabit (Gbit)} = 2^{30} \approx 10^9 \text{ biți};$$

$$1 \text{ Terabit (Tbit)} = 2^{40} \approx 10^{12} \text{ biți};$$

$$1 \text{ Petabit (Pbit)} = 2^{50} \approx 10^{15} \text{ biți.}$$

Întrebări și exerciții

- ❶ Cum se definește o sursă de informație? Dați câteva exemple.
- ❷ Care este destinația canalului de transmisie?
- ❸ Cum se determină cantitatea de informație dintr-un mesaj? Dar din N mesaje?
- ❹ Care este unitatea de măsură a informației și ce semnificație are ea?
- ❺ Determinați cantitatea de informație într-un mesaj al surselor cu următoarele mesaje posibile:
 - a) literele mari și mici ale alfabetului latin;
 - b) literele mari și mici ale alfabetului grec;
 - c) literele mari și mici ale alfabetului român;
 - d) cifrele zecimale 0, 1, 2, ..., 9;
 - e) cifrele 0, 1, 2, ..., 9, semnele +, −, ×, / și parantezele ();
 - f) indicațiile numerice de forma $hh:mm$ (hh – ora, mm – minutele) ale unui ceas electronic;
 - g) indicațiile numerice de forma $hh:mm:ss$ (ss – secunde) ale unui ceas electronic;
 - h) indicațiile numerice de forma $zz.ll.aa$ (zz – ziua, ll – luna, aa – anul) ale unui calendar electronic.
- ❻ Pentru fiecare dintre sursele indicate în exercițiul 5 determinați cantitatea de informație ce este conținută în 1 000 de mesaje emise de sursă.
- ❼ Elaborați un program care calculează cantitatea de informație din N mesaje emise de o sursă cu n mesaje posibile.

2.2. Codificarea și decodificarea informației

Se numește **semn** un element al unei mulțimi finite de obiecte ce se pot distinge.

O mulțime de semne ordonate liniar se numește **alfabet**.

Prezentăm în continuare unele dintre nenumăratele alfabetele folosite de oameni:

- a) alfabetul cifrelor zecimale: 0, 1, 2, ..., 9;
- b) alfabetul literelor latine mari: A, B, C, ..., Z;
- c) mulțimea semnelor zodiacului;
- d) mulțimea fazelor lunii.

O importanță deosebită o au alfabetele de numai două semne. Aceste alfabetele se numesc **alfabete binare**, iar semnele respective – **semne binare**.

Prezentăm câteva exemple de alfabet binare:

- a) cifrele {0, 1};
- b) perechea de culori {roșu, galben};
- c) perechea de stări {închis, deschis};
- d) perechea de răspunsuri {da, nu};
- e) perechea de tensiuni {0V, 2V};
- f) perechea de stări {magnetizat, nemagnetizat};
- g) perechea de semne {+, -} etc.

S-a convenit ca semnele unui alfabet binar să fie reprezentate prin cifrele {0, 1}, denumite **cifre binare** (*binary digit*).

Un șir finit din m semne, dintre care unele se pot repeta, formează un **cuvânt**, m reprezentând **lungimea cuvântului**. Cuvintele formate din semne binare se numesc **cuvinte binare**. Evident, cuvintele pot avea lungime variabilă sau constantă. În ultimul caz ele se numesc **cuvinte m -poziționale**. În continuare prezentăm unele mulțimi de cuvinte cu lungime constantă:

- 1-poziționale: {0, 1};
- 2-poziționale: {00, 01, 10, 11};
- 3-poziționale: {000, 001, 010, 011, 100, 101, 110, 111};
- 4-poziționale: {0000, 0001, ..., 1110, 1111}.

Se observă că cuvintele $(m + 1)$ -poziționale se formează câte două din cuvintele m -poziționale prin adăugarea cifrelor binare 0 și 1. Prin urmare, mulțimea cuvintelor m -poziționale include 2^m cuvinte distincte.

Cuvintele binare se utilizează pentru reprezentarea, transmiterea, păstrarea și prelucrarea mesajelor $s_1, s_2, \dots, s_n$ ale sursei de informație (fig. 2.2).

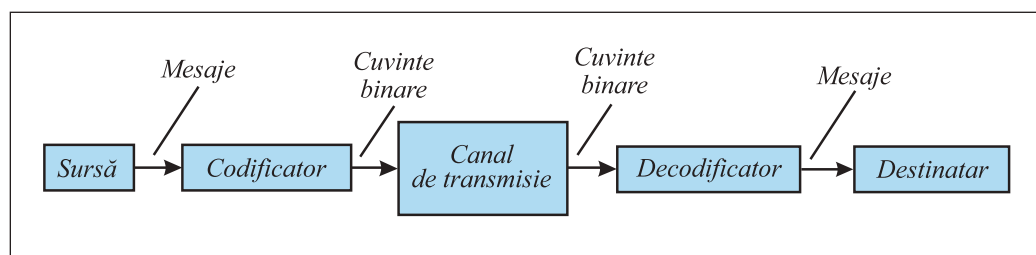


Fig. 2.2. Codificarea și decodificarea mesajelor în sistemele de transmisie a informației

Regula de transformare a mesajelor în cuvinte se numește cod, iar operația respectivă – codificare. Operația inversă codificării se numește decodificare. Dispozitivele tehnice care realizează operațiile în cauză se numesc, respectiv, codicator și decodicator.

Cel mai simplu este codul în care mesajelor posibile $s_1, s_2, \dots, s_n$ le corespund cuvinte binare de lungime constantă m . Acest cod denumit **cod m -pozițional** poate fi definit cu ajutorul unui tabel. În figura 2.3 sunt prezentate tabelele respective pentru surse cu $n = 2, 3, 4, \dots, 8$ mesaje posibile.

$n=2, m=1$		$n=3, m=2$		$n=4, m=2$	
s_1	0	s_1	00	s_1	00
s_2	1	s_2	01	s_2	01
		s_3	10	s_3	10
				s_4	11

$n=5, m=3$		$n=6, m=3$		$n=7, m=3$		$n=8, m=3$	
s_1	000	s_1	000	s_1	000	s_1	000
s_2	001	s_2	001	s_2	001	s_2	001
s_3	010	s_3	010	s_3	010	s_3	010
s_4	011	s_4	011	s_4	011	s_4	011
s_5	100	s_5	100	s_5	100	s_5	100
		s_6	101	s_6	101	s_6	101
				s_7	110	s_7	110
						s_8	111

Fig. 2.3. Coduri de cuvinte cu lungime constantă (coduri m -poziționale)

Operațiile de codificare și decodificare constau în extragerea datelor necesare din tabel. Evident, decodificarea va fi univocă numai când cuvintele binare incluse în tabel sunt distincte. Acest lucru este posibil dacă lungimea m a cuvintelor de cod satisface inegalitatea

$$2^m \geq n.$$

După logaritmare obținem:

$$m \geq \log_2 n.$$

Întrucât expresia $\log_2 n$ exprimă cantitatea de informație, se poate afirma:

Lungimea cuvintelor unui cod pozițional trebuie să fie mai mare sau egală cu cantitatea de informație a unui mesaj.

De exemplu, lungimea cuvintelor pentru codificarea literelor mari ale alfabetului latin $\{A, B, C, \dots, Z\}$, $n = 26$, se determină din relația

$$m \geq \log_2 26 \approx 4,700.$$

Stabilind $m = 5$, putem forma cuvintele binare ale codului 5-pozițional:

$A - 00000$

$B - 00001$

$C - 00010$

$D - 00011$

$E - 00100$

...

$Z - 11001.$

Un astfel de cod a fost propus de filosoful și omul de stat englez Francis Bacon încă în anul 1580.

Algoritmii de elaborare a **codurilor cu cuvinte de lungime variabilă** sunt mult mai complicați și se studiază în cursurile avansate de informatică.

Întrebări și exerciții

- ❶ Ce este un alfabet? Dați exemple de alfabet binare.
- ❷ Cum se reprezintă semnele oricărui alfabet binar?
- ❸ Explicați cum pot fi formate cuvintele binare $(m+1)$ -poziționale. Care este numărul cuvintelor binare m -poziționale distincte?
- ❹ Care este destinația unui cod? Cum se definește codul m -pozițional?
- ❺ Cum se efectuează codificarea și decodificarea mesajelor în cazurile când codul este definit printr-un tabel?
- ❻ Codificați mesajele s_3, s_4 și s_6 ale unei surse cu 7 mesaje posibile. Utilizați codul 3-pozițional din *figura 2.3*.
- ❼ Decodificați mesajele 100, 000 și 010, reprezentate în codul 3-pozițional din *figura 2.3*, $n = 5$.
- ❽ Cum se determină numărul de semne binare necesare pentru formarea cuvintelor de lungime constantă ale unui cod?
- ❾ Utilizând codul 3-pozițional din *figura 2.3*, $n = 6$, codificați șirul de mesaje $s_1, s_2, s_6, s_5, s_3, s_6, s_3, s_2, s_1$.
- ❿ Cum influențează cantitatea de informație a unui mesaj asupra lungimii cuvintelor de cod?
- ⓫ Explicați sensul termenilor **cantitate de informație** și **informație**.
- ⓬ Elaborați un program care codifică și decodifică literele alfabetului latin. Se va utiliza codul propus de Francis Bacon.
- ⓭ Elaborați un program care alcătuiește tabelul unui cod m -pozițional pentru o sursă cu n mesaje posibile.

2.3. Coduri frecvent utilizate

Orice cod, utilizat pentru reprezentarea, transmiterea, păstrarea și prelucrarea informației, trebuie să fie econom și insensibil la perturbații, iar echipamentele respective de codificare și decodificare – să fie simple. Pe parcursul dezvoltării tehnicii de calcul au fost elaborate mai multe coduri. Aceste coduri se clasifică în coduri numerice și coduri alfanumerice.

Codurile numerice oferă posibilitatea reprezentării cifrelor $\{0, 1, 2, \dots, 9\}$ prin cuvinte binare 4-poziționale. Exemple de coduri numerice sunt prezentate în *tabelul 2.2*, p. 98.

Coduri numerice

Cifra	Denumirea codului			
	<i>Direct</i>	<i>Gray</i>	<i>Aiken</i>	<i>Exces 3</i>
0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0010	0011	0010	0101
3	0011	0010	0011	0110
4	0100	0110	0100	0111
5	0101	0111	1011	1000
6	0110	0101	1100	1001
7	0111	0100	1101	1010
8	1000	1100	1110	1011
9	1001	1101	1111	1100

Codurile alfanumerice reprezintă prin cuvinte binare cifrele 0, 1, 2, ..., 9, literele mari și mici ale alfabetului, semnele de punctuație, semnele operațiilor aritmetice etc. În *tabelul 2.3* este prezentat codul **ASCII** (*American Standard Code for Information Interchange*), inventat în anul 1968.

Tabelul 2.3

Codul ASCII

Simbol	Cuvânt binar	Echivalent zecimal	Simbol	Cuvânt binar	Echivalent zecimal
Spațiu	0100000	32	P	1010000	80
!	0100001	33	Q	1010001	81
"	0100010	34	R	1010010	82
#	0100011	35	S	1010011	83
\$	0100100	36	T	1010100	84
%	0100101	37	U	1010101	85
&	0100110	38	V	1010110	86
'	0100111	39	W	1010111	87
(	0101000	40	X	1011000	88
)	0101001	41	Y	1011001	89
*	0101010	42	Z	1011010	90
+	0101011	43	[	1011011	91
,	0101100	44	\	1011100	92
-	0101101	45	]	1011101	93
.	0101110	46	^	1011110	94

Simbol	Cuvânt binar	Echivalent zecimal	Simbol	Cuvânt binar	Echivalent zecimal
/	0101111	47	—	1011111	95
0	0110000	48	`	1100000	96
1	0110001	49	a	1100001	97
2	0110010	50	b	1100010	98
3	0110011	51	c	1100011	99
4	0110100	52	d	1100100	100
5	0110101	53	e	1100101	101
6	0110110	54	f	1100110	102
7	0110111	55	g	1100111	103
8	0111000	56	h	1101000	104
9	0111001	57	i	1101001	105
:	0111010	58	j	1101010	106
;	0111011	59	k	1101011	107
<	0111100	60	l	1101100	108
=	0111101	61	m	1101101	109
>	0111110	62	n	1101110	110
?	0111111	63	o	1101111	111
@	1000000	64	p	1110000	112
A	1000001	65	q	1110001	113
B	1000010	66	r	1110010	114
C	1000011	67	s	1110011	115
D	1000100	68	t	1110100	116
E	1000101	69	u	1110101	117
F	1000110	70	v	1110110	118
G	1000111	71	w	1110111	119
H	1001000	72	x	1111000	120
I	1001001	73	y	1111001	121
J	1001010	74	z	1111010	122
K	1001011	75	{	1111011	123
L	1001100	76		1111100	124
M	1001101	77	}	1111101	125
N	1001110	78	~	1111110	126
O	1001111	79	Del	1111111	127

Acest cod este 7-pozițional și include $2^7 = 128$ de simboluri. Primele 32 de simboluri (cuvintele binare 0000000, 0000001, 0000010, ..., 0011111) specifică detaliile tehnice ale transmisiunilor de informații și nu au fost incluse în tabel.

Cuvintele binare 0100000, 0100001, 0100010, ..., 1111110 reprezintă caracterele imprimabile din textele în limba engleză. Cuvântul 1111111 reprezintă caracterul neimprimabil *Delete* (Anulare).

Codificarea mesajelor se realizează prin înlocuirea simbolurilor cu cuvintele binare respective. De exemplu, cuvântul START se reprezintă în codul ASCII prin următoarea secvență de cuvinte binare:

1010011 1010100 1000001 1010010 1010100.

Evident, decodificarea se va realiza în ordine inversă. De exemplu, secvența de cuvinte binare

1010011 1010100 1001111 1010000

reprezintă în codul ASCII cuvântul STOP.

De regulă, limbajele de programare operează nu cu cuvintele binare propriu-zise, dar cu echivalentele lor zecimale. În programele PASCAL echivalentele zecimale ale caracterelor pot fi aflate cu ajutorul funcției predefinite `ord`. De exemplu:

`ord('S')=83;` `ord('T')=84;` `ord('A')=65;` `ord('R')=82`

etc. Funcția predefinită `chr` returnează caracterul care corespunde echivalentului zecimal indicat. Astfel,

`chr(83)='S';` `chr(84)='T';` `chr(65)='A';` `chr(82)='R'.`

Într-un mod similar, în programele C++ scriem:

`int('S')=83;` `int('T')=84;` `int('A')=65;` `int('R')=82`
`char(83)='S';` `char(84)='T';` `char(65)='A';` `char(82)='R'.`

Orientat la textele engleze, codul ASCII nu include literele cu semne diacritice și caracterele grafice speciale întâlnite în diferite limbi europene și în lucrările științifice. De aceea pentru calculatoarele moderne s-au elaborat versiuni dedicate ale codului ASCII, denumite **coduri ASCII extinse**. Codurile extinse sunt 8-poziționale și includ $2^8 = 256$ de simboluri. Structura codurilor respective este prezentată în *tabelul 2.4*.

Tabelul 2.4

Structura codurilor ASCII extinse

Simbol	Cuvânt binar	Echivalent zecimal	Observații
Spațiu	00100000	32	Partea 1: – simbolurile codului ASCII
!	00100001	33	
"	00100010	34	
#	00100011	35	
...	...	...	
}	01111101	125	
~	01111110	126	
Del	01111111	127	

Simbol	Cuvânt binar	Echivalent zecimal	Observații
A	10000000	128	Partea 2: – simboluri specifice limbilor naționale; – caractere grafice; – caractere științifice
Б	10000001	129	
B	10000010	130	
...	...	...	
≡	11110000	240	
Ă	11110001	241	
ă	11110010	242	
Â	11110011	243	
â	11110100	244	
Î	11110101	245	
î	11110110	246	
Ș	11110111	247	
ș	11111000	248	
'	11111001	249	
—	11111010	250	
√	11111011	251	
ƒ	11111100	252	
ț	11111101	253	
□	11111110	254	
	11111111	255	

Partea 1 a fiecărui cod extins include simbolurile de la 0 la 127 oferite de codul ASCII. **Partea a 2-a** este specifică fiecărei țări și include simbolurile de la 128 la 255. Aceste simboluri sunt utilizate pentru reprezentarea literelor alfabetelor naționale, precum și a caracterelor științifice frecvent utilizate. Pentru exemplificare, în *tabelul 2.4* sunt prezentate codurile literelor Ă, ă, Â, â, Î, î, Ș, ș, ƒ, ț din alfabetul limbii române propuse în anul 1992 de firma *TISH* (Chișinău). Este firesc ca utilizarea codurilor extinse să asigure prelucrarea informațiilor prezentate în diferite limbi.

Un alt exemplu de cod alfanumeric este codul binar 8-pozițional **EBCDIC** (*Extended Binary Coded Data Interchange Code*), care se utilizează pe calculatoarele mari.

Menționăm că extinderea domeniului de aplicare a codurilor 8-poziționale a favorizat utilizarea octetului și a multiplilor lui pentru măsurarea cantității de informație:

$$1 \text{ octet} = 2^3 = 8 \text{ biți};$$

$$1 \text{ Kiloctet} = 2^{10} \approx 10^3 \text{ octeți};$$

$$1 \text{ Megaoctet} = 2^{20} \approx 10^6 \text{ octeți};$$

$$1 \text{ Gigaoctet} = 2^{30} \approx 10^9 \text{ octeți};$$

$$1 \text{ Teraoctet} = 2^{40} \approx 10^{12} \text{ octeți};$$

$$1 \text{ Petaoctet} = 2^{50} \approx 10^{15} \text{ octeți}.$$

În literatura de specialitate octetul este notat prin *B* (*byte*), iar multiplii respectivi prin *KB*, *MB*, *GB*, *TB* și *PB*.

Este cunoscut faptul că informatica este supusă într-o măsură foarte mare fenomenului globalizării, produsele-program și echipamentele digitale fiind concepute în așa mod, încât ele pot procesa informații prezentate în diverse limbi. Pentru reprezentarea pe calculator a caracterelor din aproape toate limbile globului, cel mai des se utilizează codul UNICODE, lungimea cuvintelor căruia poate ajunge până la 32 de cifre binare.

Întrebări și exerciții

- ❶ Câte mesaje posibile pot fi codificate cu ajutorul unui cod m -pozițional?
- ❷ Determinați codurile admise de sistemul de operare cu care lucrați dvs.
- ❸ Codificați în codul *Gray* următoarele șiruri de cifre zecimale: 123, 461, 952, 783, 472.
- ❹ Decodificați mesaje reprezentate în codul *Aiken*:

a) 0011 1111 0100	d) 1110 0010 1101
b) 1111 0000 0100	e) 0011 1100 1111
c) 0010 0001 1011	f) 1111 1101 0000
- ❺ Codificați în codul ASCII expresiile:

a) A+B	d) NEXT I
b) FOR I=1 TO N	e) PAUSE
c) PRINT A\$	f) PROGRAM
- ❻ Decodificați mesaje reprezentate în codul ASCII:

a) 1000010	1100101	1100111	1101001	1101110;
b) 1010011	1110100	1101111	1110000;	
c) 1000101	1101110	1100100;		
d) 1101001	0111010	0111101	0110001	0111011.
- ❼ Elaborați un program care afișează pe ecran codurile următoarelor caractere, introduse de la tastatură:
 - a) cifrele zecimale 0, 1, 2, ..., 9;
 - b) literele latine mari A, B, C, ..., Z;
 - c) literele latine mici a, b, c, ..., z;
 - d) semnele operațiilor aritmetice;
 - e) caracterele speciale ;, <, =, >, ?, [,], {, }, /, \.
- ❽ STUDIU DE CAZ. Cu ajutorul unui motor de căutare, găsiți pe Internet descrierile codurilor frecvent utilizate. Stabiliți parcursul istoric al codurilor respective, avantajele și neajunsurile acestora, domeniile lor de utilizare.

2.4. Informația mesajelor continue

Sursele de informație studiate până în prezent se definesc printr-o variabilă S , care poate lua valori dintr-o mulțime finită de elemente distincte $\{s_1, s_2, \dots, s_n\}$, denumită mulțimea mesajelor posibile. Practica ne demonstrează că nu toate sursele de informație pot fi definite direct în acest mod. Drept exemplu vom remarca termometrele cu mercur sau alcool, vitezometrele automobilelor, microfoanele, camerele de luat vederi etc. Astfel de surse pot fi definite printr-o variabilă S (temperatura, viteza momentană, tensiunea la bornele de ieșire ale microfonului etc.) care poate lua orice valoare într-un anumit interval $[s_{\min}, s_{\max}]$.

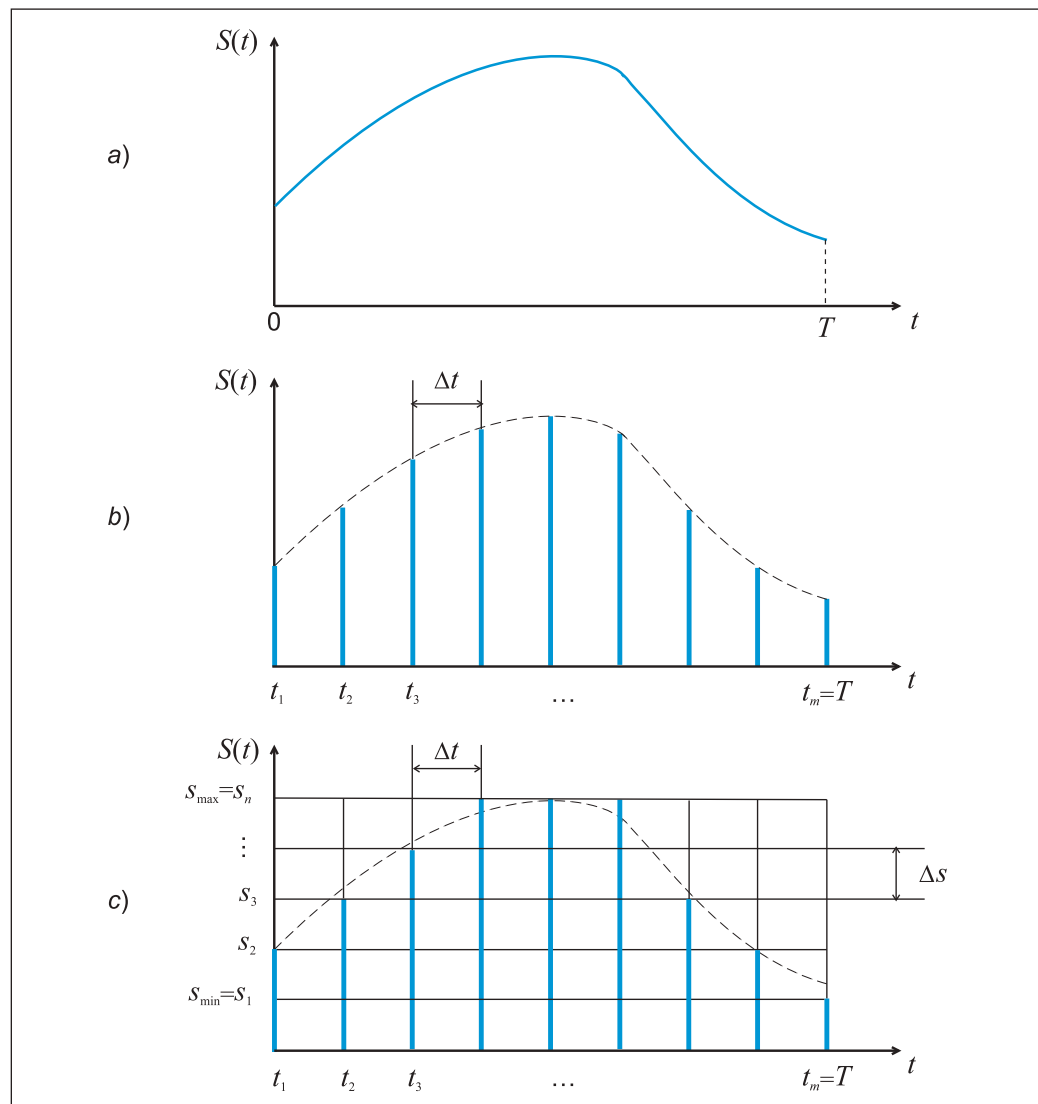


Fig. 2.4. Discretizarea mesajelor continue: a – mesaj continuu; b – mesaj discretizat în timp (eșantionat); c – mesaj eșantionat și discretizat în valoare (cuantificat)

Sursele de informație care sunt definite printr-o variabilă S ce ia valori dintr-o mulțime finită de elemente distincte se numesc **surse cu mesaje discrete**. Sursele care se definesc printr-o variabilă S ce poate lua orice valori într-un anumit interval se numesc **surse cu mesaje continue**.

Ca și în cazul surselor discrete, mesajele continue se produc în timp. Prin urmare, S este o funcție de timp, $S = S(t)$. În scopul evaluării cantității de informație în mesajele continue, vom analiza valorile funcției $S(t)$ numai în momentele de timp $t_1, t_2, \dots, t_m$ (fig. 2.4). Valorile date, notate prin $S(t_1), S(t_2), \dots, S(t_m)$, se numesc **eșantioane** (din franceză *echantillon* „cantitate mică luată dintr-un produs pentru a determina calitatea lui”).

De obicei, momentele de timp $t_1, t_2, \dots, t_m$ se definesc conform relației:

$$t_i = t_{i-1} + \Delta t.$$

Mărimea Δt se numește **perioada de eșantionare**. Valoarea concretă a perioadei de eșantionare Δt este determinată de viteza cu care $S(t)$ variază în timp.

De exemplu, în meteorologie schimbările de temperatură se produc pe parcurs de ore și $\Delta t = 1$ h, iar în tehnica prelucrării semnalelor sonore $\Delta t = 5 \cdot 10^{-5}$ s.

Operația de transformare a mesajelor continue în eșantioane se numește discretizare în timp sau eșantionare.

Evident, până la recepția mesajului continuu, valorile concrete ale eșantioanelor $S(t_1), S(t_2), \dots, S(t_m)$ sunt necunoscute destinatarului. Se cunoaște numai intervalul $[s_{\min}, s_{\max}]$ în care poate lua valori arbitrare orice eșantion. În scopul evaluării cantității de informație dintr-un eșantion, vom rotunji valoarea $S(t_i)$ la una din valorile prestabilite $s_1, s_2, \dots, s_n$ (fig. 2.4).

Valorile prestabilite $s_1, s_2, \dots, s_n$ se numesc **cuante**, iar operația de transformare a valorilor curente ale mesajelor continue în cuante se numește **discretizare în valoare sau cuantificare**.

De obicei,

$$s_1 = s_{\min}, \quad s_2 = s_{\min} + \Delta s, \quad \dots, \quad s_i = s_{i-1} + \Delta s, \quad \dots, \quad s_n = s_{\max}.$$

Mărimea Δs reprezintă **pasul** sau **intervalul de cuantificare**. Valoarea concretă a intervalului de cuantificare depinde de natura fizică a sursei de informație, precizia de măsurare, puterea de rezoluție a destinatarului etc.

De exemplu, pentru un termometru medical $s_{\min} = 34^\circ$, $s_{\max} = 42^\circ$ și $\Delta s = 0,1^\circ$. În meteorologie $s_{\min} = -60^\circ$, $s_{\max} = +60^\circ$, $\Delta s = 1^\circ$. Pentru vitezometrul unui automobil $s_{\min} = 0$, $s_{\max} = 150$ km/h, $\Delta s = 5$ km/h.

Numărul eșantioanelor m și numărul cuantelor n se determină din următoarele relații (fig. 2.4 b și c):

$$m = \frac{T}{\Delta t} + 1; \quad n = \frac{|s_{\max} - s_{\min}|}{\Delta s} + 1,$$

unde T este durata mesajului continuu.

Întrucât cuantele $s_1, s_2, \dots, s_n$ pot fi considerate mesaje discrete, cantitatea de informație într-un eșantion

$$I = \log_2 n = \log_2 \left(\frac{|s_{\max} - s_{\min}|}{\Delta s} + 1 \right),$$

iar cantitatea de informație într-un mesaj continuu:

$$V = mI = \left(\frac{T}{\Delta t} + 1 \right) \log_2 \left(\frac{|s_{\max} - s_{\min}|}{\Delta s} + 1 \right).$$

De exemplu, pentru termometrul medical

$$I = \log_2 \left(\frac{|42 - 34|}{0,1} + 1 \right) \approx 6,34 \text{ biți},$$

iar pentru vitezometrul automobilului

$$I = \log_2 \left(\frac{|150 - 0|}{5} + 1 \right) \approx 4,95 \text{ biți}.$$

Cantitatea de informație dintr-o înregistrare audio, pentru care $n = 256$, $\Delta t = 5 \cdot 10^{-5}$ s și $T = 45$ min. este

$$V = \frac{45 \cdot 60}{5 \cdot 10^{-5}} \log_2 256 = 4,32 \cdot 10^8 \text{ biți} = 432 \text{ Mbiți}.$$

Dacă precizia de măsurare și puterea de rezoluție a destinatarului cresc, perioada de eșantionare Δt și pasul de cuantificare Δs pot fi micșorate. În consecință, va crește și cantitatea de informație dintr-un mesaj continuu.

Informația mesajelor continue poate fi reprezentată printr-un set de cuvinte binare. Pentru aceasta, cuantele $s_1, s_2, \dots, s_n$ se codifică exact la fel ca și oricare alte mesaje discrete. De exemplu, indicațiile unui termometru medical ($I \approx 6,34$ biți) pot fi codificate cu un cod 7-pozițional. Cel mai frecvent se utilizează codurile numerice directe (tab. 2.2), cuvântul de cod reprezentând numărul cuantei respective. În unele aplicații se utilizează codul *Gray*, care este insensibil la perturbații.

Dispozitivul care transformă mesajul continuu aplicat la intrare într-o succesiune de cuvinte de cod se numește **convertor analog-numeric**. Operația inversă, și anume transformarea cuvintelor de cod aplicate la intrare în valorile cuantelor respective, se efectuează cu ajutorul **convertoarelor numeric-analogice**. Utilizarea convertoarelor este necesară în cazurile în care informația de procesat este reprezentată prin mesaje continue: controlul proceselor tehnologice, dirijarea obiectelor aflate în mișcare, monitorizarea parametrilor fiziologici în medicină, filtrarea și mixajul semnalelor audio etc.

Întrebări și exerciții

- 1 Care este diferența dintre sursele cu mesaje discrete și sursele cu mesaje continue?
- 2 Dați câteva exemple de surse cu mesaje continue. Concretizați intervalul în care ia valori variabila ce definește sursa.
- 3 Explicați cum se efectuează operația de eșantionare. Cum se alege perioada de eșantionare?
- 4 Explicați cum se efectuează operația de cuantificare. Cum se alege pasul de cuantificare?

- ⑤ Cum influențează valorile perioadei de eșantionare și ale pasului de cuantificare cantitatea de informație extrasă dintr-un mesaj continuu?
- ⑥ Altimetrul cu impulsuri al unui avion poate măsura înălțimi de la 100 m până la 20 km. Eroarea de măsurare nu depășește 1 m. Pentru a efectua o măsurare sunt necesare 10^{-3} s. Determinați cantitatea de informație furnizată de altimetru timp de 5 ore de zbor.
- ⑦ Temperatura din interiorul unui reactor chimic se înregistrează pe o bandă de hârtie milimetrică. Pe axa absciselor se indică timpul (1 mm reprezintă o oră), iar pe axa ordonatelor – temperatura (1 mm reprezintă 10°C). Câtă informație conține o înregistrare efectuată timp de 30 de zile, dacă temperatura poate varia de la 80 până la 1000°C ?
- ⑧ Pentru înregistrarea sunetului se utilizează microfoane, tensiunea de ieșire a cărora variază de la 0 până la $100\ \mu\text{V}$. Aparatul de înregistrare nu distinge tensiunile valorile cărora diferă cu mai puțin de $0,1\ \mu\text{V}$. Pentru a asigura o reproducere fidelă, în fiecare secundă se iau 40 000 de eșantioane. Câtă informație va furniza microfonul dat timp de 3 ore?
- ⑨ Care este destinația convertoarelor analog-numerice și numeric-analogice?
- ⑩ Elaborați un program care introduce de la tastatură valorile curente ale eșantioanelor și afișează pe ecran codurile cuantelor corespunzătoare.
- ⑪ Elaborați un program care simulează funcționarea unui convertor numeric-analog.
- ⑫ CERCETEAZĂ! Utilizând un motor de căutare, găsiți pe Internet descrierea aparatelor de înregistrare digitală a sunetelor. Aflați perioada de eșantionare și pasul de cuantificare utilizate în digitalizarea mesajelor sonore, realizată de fiecare dintre aparatele respective.
- ⑬ STUDIU DE CAZ. Cu ajutorul unui motor de căutare, găsiți pe Internet descrierile aparatelor de înregistrare digitală a sunetelor, propuse de unitățile de comerț. Comparați aceste aparate din perspectiva modului de digitalizare a mesajelor sonore. Aflați cum parametrii de discretizare influențează costurile aparatelor de înregistrare a sunetelor, care sunt avantajele, neajunsurile și domeniile de utilizare a acestora.

2.5. Cuantizarea imaginilor

Imagine se numește reprezentarea unui obiect, executată pe o suprafață prin acțiunea directă a utilizatorului sau prin intermediul unui echipament. Cu titlul de exemplu amintim desenele, fotografiile, imaginile formate de diverse sisteme optice, optico-mecanice sau optico-electronice: microscopul, telescopul, aparatele cinematografice, televiziunea etc.

Pentru a evalua cantitatea de informație, imaginea este împărțită în **microzone**, numite de cele mai multe ori **puncte** sau **pixeli**. Descompunerea imaginii în puncte se realizează cu ajutorul unui **rastru** (de la cuvântul latin *raster* „greblă”). Rastrul reprezintă o suprafață plană, în general dreptunghiulară, pe care sunt trasate două seturi de linii paralele, perpendiculare între ele (*fig. 2.5*). Densitatea

liniilor și, respectiv, densitatea punctelor caracterizează puterea de rezoluție a echipamentelor pentru reproducerea sau formarea imaginilor.

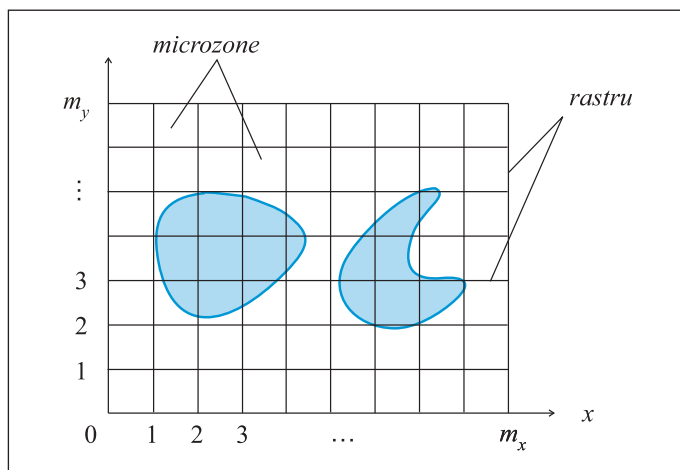


Fig. 2.5. Descompunerea imaginii în microzone

De exemplu, pentru ilustrațiile de gazetă se folosesc rastre cu rezoluția 24-30 linii/cm (576-900 de puncte pe 1 cm^2), iar pentru reproducerea tablourilor – rastre cu 54-60 linii/cm. Rastrul vizualizatorului, adică desenul pe care-l formează fasciculul de electroni pe ecranul tubului catodic, poate include 640×480 , 800×600 , 720×400 , ..., 1024×1024 de puncte.

Descompunerea imaginii în puncte (microzone) reprezintă o procedură de discretizare în spațiu.

În cazul imaginilor monocrome (alb-negru), fiecare microzonă se descrie prin **luminanța** (strălucirea) sa, care în general este o mărime continuă. Această mărime poate fi discretizată în valoare (cuantificată). Numărul cuantelor n va caracteriza puterea de rezoluție a echipamentelor pentru reproducerea sau formarea imaginilor. Prin urmare, cantitatea de informație a unei imagini monocrome:

$$I = m_x m_y \log_2 n,$$

unde m_x și m_y reprezintă numărul de microzone ale rastrului respectiv pe orizontală și verticală (fig. 2.5).

Întrucât culorile pot fi redată prin suprapunerea a trei reprezentări ale aceleiași imagini în roșu, verde și albastru, cantitatea de informație dintr-o imagine color se determină din relația:

$$I = 3 m_x m_y \log_2 n.$$

Imaginile obiectelor în mișcare se discretizează în timp, de obicei 24 (cinematograful) sau 25 (televizorul) de cadre pe secundă. Prin urmare, cantitatea de informație a unui film cu durată T se determină din relația:

$$V = T f I,$$

unde f este frecvența cadrelor, iar I cantitatea de informație dintr-un singur cadru.

De exemplu, în televiziune $m_x \approx m_y = 625$, $n = 32$ și $f = 25$ de cadre pe secundă. Un cadru color va conține:

$$I = 3 \cdot 625 \cdot 625 \cdot \log_2 32 \approx 5,6 \text{ Mbiți.}$$

Un film color cu durată de 1,5 ore va conține:

$$V = 1,5 \cdot 3\,600 \cdot 25 \cdot I \approx 791 \text{ Gbiți.}$$

Setul de cuvinte binare care reprezintă informația microzonelor se numește imagine numerică. Operația de transformare a imaginii într-un set de cuvinte binare se numește cuantizarea imaginii.

Imaginile preluate de camerele video se cuantizează cu ajutorul convertoarelor analog-numeric. Imaginile de pe hârtie pot fi cuantizate cu ajutorul unui dispozitiv special, numit **scaner**. Acest dispozitiv conține celule fotosensibile, convertoare analog-numeric și mecanisme de avansare a hârtiei.

Imaginile numerice se transformă în imagini propriu-zise cu ajutorul convertoarelor numeric-analogice și al echipamentelor de formare a rastrului: tubul catodic și sistemul de baleiere în vizualizatoare, matricea de ace în imprimantele mecanice etc.

Întrebări și exerciții

- 1 Numiți operațiile necesare pentru a cuantiza imaginea.
- 2 Care este destinația rastrului? Din care considerente se alege densitatea liniilor unui rastru?
- 3 Cum se evaluează cantitatea de informație dintr-o imagine monocromă?
- 4 Cum pot fi redată culorile unei imagini multicolore? Cum se evaluează cantitatea de informație dintr-o imagine color?
- 5 Evaluați cantitatea de informație dintr-o fotografie de ziar cu dimensiunile 10×10 cm, redată cu ajutorul unui rastru ce conține 24 de puncte/cm. Fiecare punct poate avea următoarele nuanțe: alb, gri-deschis, gri-închis, negru.
- 6 Câtă informație conține o fotografie color cu dimensiunile 20×20 cm, reproducă cu ajutorul unui rastru ce conține 60 de puncte/cm? Pot fi redată până la 256 de niveluri de luminanță ale punctelor respective.
- 7 Rastrul unei camere de luat vederi este format din 1024×1024 de puncte. Pot fi redată până la 64 de niveluri de luminanță ale punctelor respective. Câtă informație va conține un film video cu durată de 3 ore?
- 8 Imaginea numerică conține câte un cuvânt binar pentru fiecare punct al rastrului unui vizualizator. Câte niveluri de luminanță pot fi redată pe ecran dacă cuvintele imaginii numerice sunt 3-poziționale? 5-poziționale? 8-poziționale?
- 9 CERCETEAZĂ! Utilizând un motor de căutare, găsiți pe Internet descrierea aparatelor digitale fotografice. Aflați puterea de rezoluție realizată de fiecare dintre aparatele respective în scopul digitalizării imaginilor statice.
- 10 STUDIU DE CAZ. Cu ajutorul unui motor de căutare, găsiți pe Internet descrierile aparatelor fotografice digitale, propuse de unitățile de comerț. Comparați aceste aparate din perspectiva modului de digitalizare a imaginilor statice.

Află cum parametrii de discretizare influențează costurile, avantajele, neajunsurile și domeniile de utilizare a aparatelor fotografice digitale.

- ⑩ **CERCETEAZĂ!** Utilizând un motor de căutare, găsiți pe Internet descrierea camerelor video digitale. Aflați puterea de rezoluție realizată de fiecare dintre camerele respective în scopul digitalizării informațiilor dinamice.
- ⑪ **STUDIU DE CAZ.** Cu ajutorul unui motor de căutare, găsiți pe Internet descrierile camerelor video digitale propuse de unitățile de comerț. Comparați aceste camere din perspectiva modului de digitalizare a imaginilor dinamice. Aflați cum parametrii de discretizare influențează costurile camerelor video digitale, care sunt avantajele, neajunsurile și domeniile de utilizare a acestora.

2.6. Reprezentarea și transmiterea informației

Obiectul material folosit pentru păstrarea, transmiterea sau prelucrarea informației se numește **purtător de informație**. Deosebim purtători statici și purtători dinamici de informație.

Purtătorii statici se utilizează pentru păstrarea informației sau, cu alte cuvinte, pentru transmiterea ei în timp. Informația înregistrată pe un purtător static poate fi citită în scopul prelucrării sau utilizării ulterioare. Primii purtători statici folosiți de omenire au fost pietrele, plăcile de lut ars, papirusul. Un alt purtător static de informație îl constituie hârtia. Mesajele înregistrate pe hârtie sub formă de manuscrise, desene sau texte tipărite pot fi păstrate un timp foarte îndelungat. În calculatoare, ca purtători statici, se utilizează:

- hârtia pentru imprimantele mecanice, cu jet de cerneală, laser etc.;
- straturile active ale benzilor, ale discurilor magnetice;
- mediile reflectoare ale discurilor optice etc.

Transmiterea informației în spațiu se realizează cu ajutorul **purtătorilor dinamici**. În calitate de purtători dinamici tehnica actuală folosește:

- unde acustice în gaze (aer) sau lichide;
- tensiuni și curenți electrici;
- unde electromagnetice etc.

Întrucât transmiterea informației se produce în spațiu și timp, cel puțin o mărime fizică a purtătorului de informație trebuie să se schimbe.

*Se numește **semnal variația mărimii fizice ce asigură transmiterea mesajelor**. Caracteristica semnalului folosită pentru reprezentarea (descrierea) mesajelor se numește **parametru informațional al semnalului**.*

De exemplu, în transmisiunile radio și TV, în calitate de purtători se utilizează undele electromagnetice. Amplitudinea sau frecvența acestor unde poate varia în timp (fig. 2.6, p. 110). În primul caz, parametrul informațional îl constituie amplitudinea oscilațiilor, iar în al doilea – frecvența lor.

În calculatoarele electronice în calitate de purtător de informație se folosește, de obicei, curentul electric, tensiunea și intensitatea curentului fiind parametrii

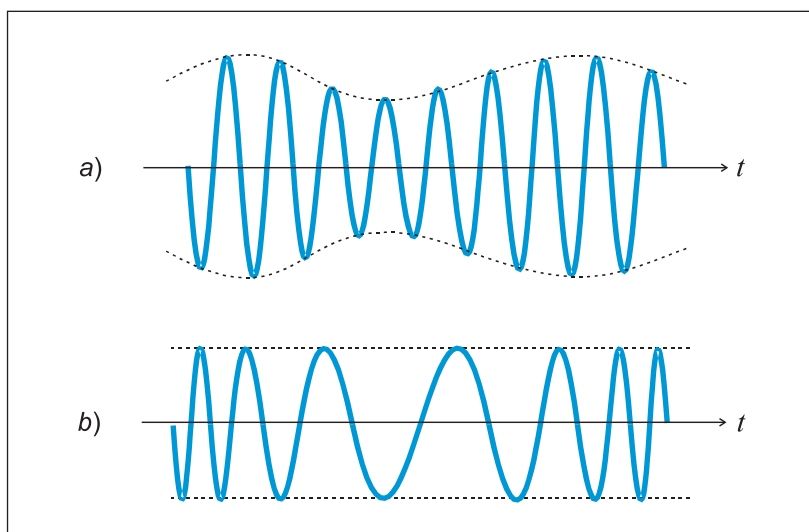


Fig. 2.6. Parametri informaționali ai undelor electromagnetice:
a – amplitudinea; b – frecvența

informaționali ai semnalului. Semnalele respective au formele prezentate în figura 2.7. În cazul nivelurilor de tensiune, unei valori a tensiunii i se asociază cifra binară 0, iar alteia cifra binară 1. Cifrele binare 0 și 1 pot fi de asemenea asociate, respectiv, cu absență sau prezență de impuls sau cu impuls negativ și impuls pozitiv.

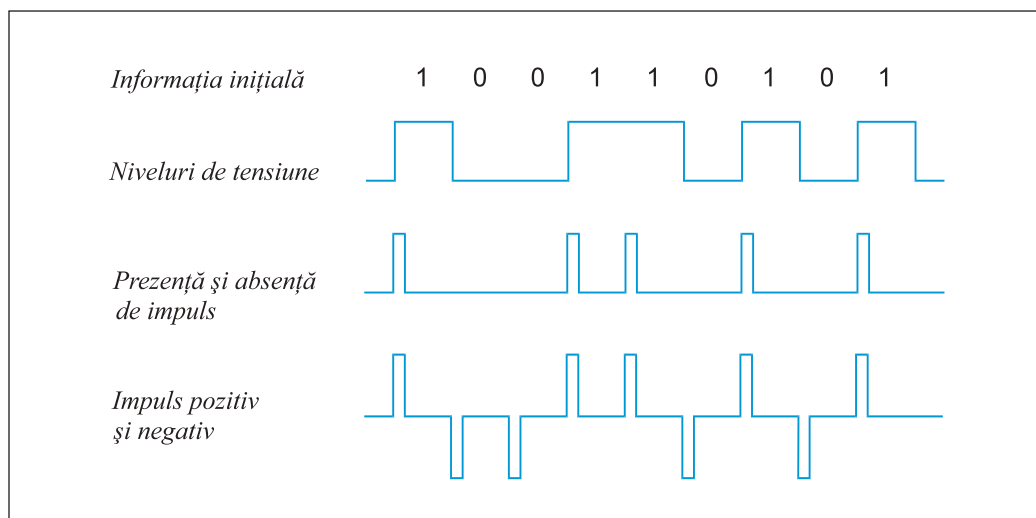


Fig. 2.7. Semnale utilizate în tehnica de calcul

Semnalul se numește discret, dacă parametrul informațional respectiv poate lua numai un număr finit de valori. Semnalul se numește continuu, dacă parametrul informațional poate lua orice valoare într-un anumit interval.

De exemplu, semnalele din *figura 2.6* sunt semnale continue, iar semnalele din *figura 2.7* sunt semnale discrete. Evident, semnalele discrete și continue sunt forme de reprezentare a mesajelor respective.

Orice sistem tehnic utilizează acele semnale care-i asigură o realizare cât mai bună a funcțiilor pentru care a fost conceput. Calculatoarele actuale utilizează niveluri de tensiune, rețelele telefonice – curenți electrici, iar radioul și televiziunea – unde electromagnetice etc. Practica demonstrează că semnalele continue pot fi transmise la distanțe mult mai mari decât cele discrete. Prin urmare, pentru a asigura legătura dintre calculatoarele aflate la distanță, la emisie semnalele discrete vor fi transformate în semnale continue. La recepție se va realiza transformarea inversă: semnalul continuu va fi transformat într-un semnal discret.

Operația prin care parametrul informațional al semnalului continuu se modifică în funcție de valorile semnalului discret se numește modulare.

Dispozitivul tehnic care realizează operația în cauză se numește **modulator**. Pentru exemplificare, în *figura 2.8* sunt prezentate operațiile de modulare în amplitudine și frecvență.

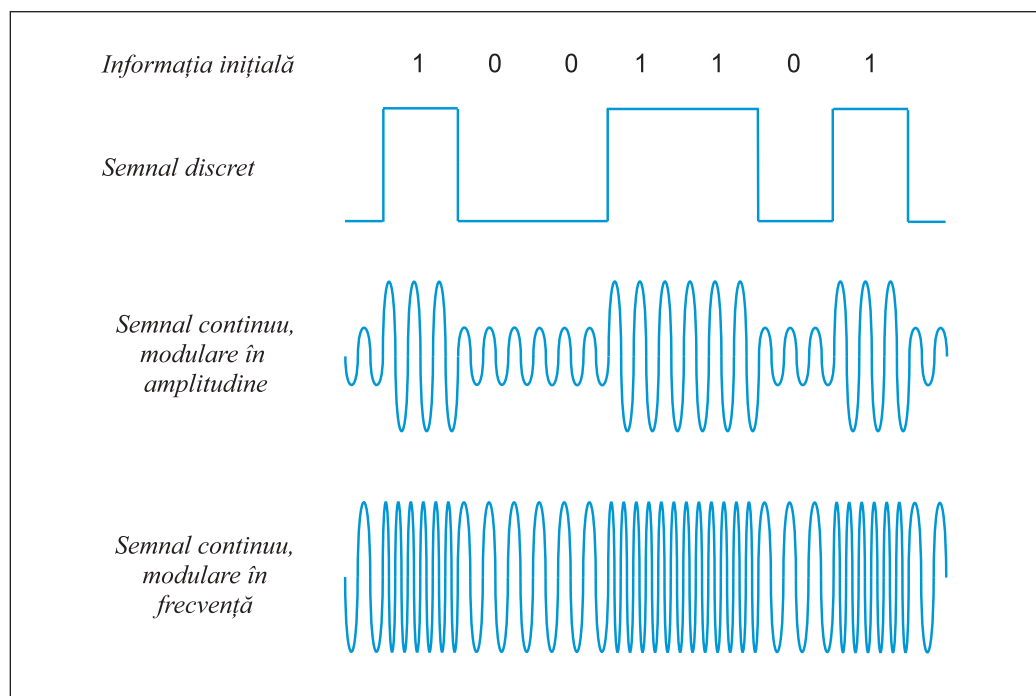


Fig. 2.8. Modularea semnalelor continue

Operația de extragere a semnalului discret dintr-un semnal continuu în funcție de procedeul de modulare adoptat se numește demodulare.

Dispozitivul tehnic care realizează operația respectivă se numește **demodulator**.

Schema detaliată a sistemului de transmisie a informației este prezentată în figura 2.9.

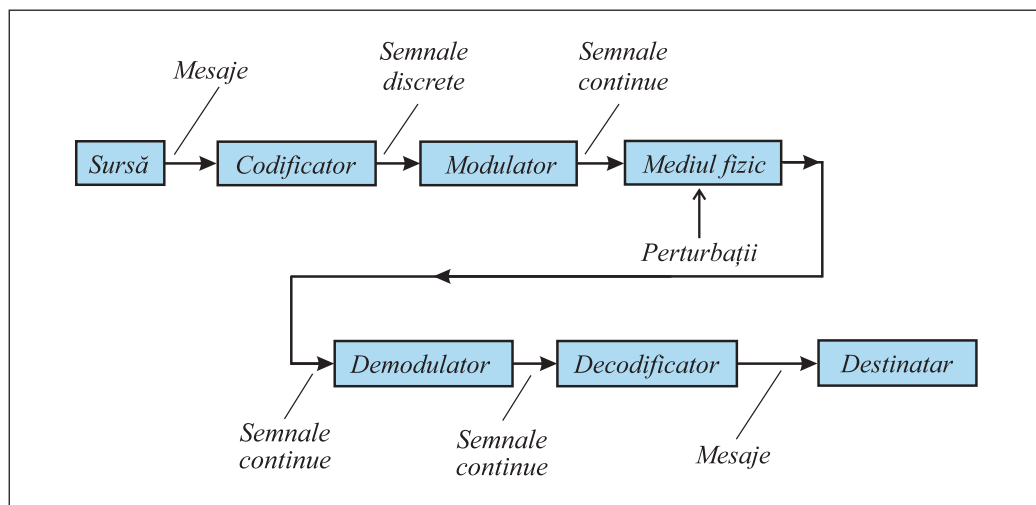


Fig. 2.9. Schema detaliată a unui sistem de transmisie a informației

Amintim destinația componentelor sistemului examinat:

codificatorul – transformă mesajele emise de sursă în cuvinte binare;

modulatorul – transformă semnalele discrete ce reprezintă cuvintele binare în semnale continue;

mediul fizic – reprezintă conductorii, fibrele optice, eterul etc. prin care se propagă semnalele continue;

demodulatorul – transformă semnalele continue în semnale discrete;

decodificatorul – transformă cuvintele binare în mesaje.

Evident, în cazul utilizării codurilor corectoare și a codurilor detectoare de erori, sistemul de transmisie va fi insensibil la perturbații.

Caracteristica principală a oricărui sistem de transmisie a informației este **capacitatea de transmisie**, exprimată în **biți pe secundă**. Capacitatea de transmisie depinde de caracteristicile fizice ale componentelor sistemului, metodele de modulare-demodulare, caracteristicile statistice ale perturbațiilor. De exemplu, capacitatea de transmisie a unui canal telefonic este de circa 34 *Kbit/s*; capacitatea unui canal radio, cu unde centimetrice – de circa 1 *Gbit/s*; capacitatea unui canal optic – de 1 *Tbit/s*.

Întrebări și exerciții

- ❶ Care este deosebirea dintre purtătorii statici și purtătorii dinamici de informație?
- ❷ Determinați tipul următorilor purtători de informație:

a) cartele perforate;	e) pelicule fotosensibile;
b) unde ultrasonore;	f) unde gravitaționale;
c) benzi perforate;	g) hârtie fotografică.
d) unde acustice;	

- ③ Numiți parametrii informaționali ai semnalelor emise de următoarele surse:
- a) microfonul;
 - b) stația de radio, unde lungi, medii sau scurte;
 - c) instrumentul muzical;
 - d) stația de radio, unde ultrascurte;
 - e) camera de luat vederi.
- ④ Descrieți purtătorii de informație și semnalele utilizate în calculatoarele actuale.
- ⑤ Explicați operațiile de modulare și demodulare a semnalelor.
- ⑥ Care este destinația modulatorului? Dar a demodulatorului?
- ⑦ În *figura 2.8* sunt redate semnalele continue ce reprezintă cuvântul binar 1001101. În codul ASCII (*tab. 2.3*) acestui cuvânt îi corespunde simbolul M. Redați pe un desen semnalele continue ce corespund următoarelor simboluri:
- | | |
|-------|-------|
| a) >; | e) <; |
| b) r; | f) K; |
| c) W; | g) a; |
| d) 9; | h) @. |
- ⑧ De ce depinde capacitatea de transmisie a canalului? În ce unități se măsoară ea?
- ⑨ Cum influențează perturbațiile asupra capacității de transmisie a canalului?
- ⑩ CERCETEAZĂ! Aflați capacitățile de transmisie a canalelor din componența sistemelor informatice la care ai acces la domiciliu, liceu, în locurile publice.

Capitolul 3

BAZELE ARITMETICE ALE TEHNICII DE CALCUL

3.1. Sisteme de numerație

În calculatoarele digitale informația de orice categorie este reprezentată, stocată și prelucrată în formă numerică. Numerele se reprezintă prin simboluri elementare denumite **cifre**.

Totalitatea regulilor de reprezentare a numerelor, împreună cu mulțimea cifrelor poartă denumirea de sistem de numerație. Numărul cifrelor definește baza sistemului de numerație.

Prezentăm câteva exemple de sisteme de numerație:

- **sistemul zecimal** este un sistem de numerație în baza 10, numărul de cifre utilizate fiind 10, respectiv, 0, 1, 2, ..., 9;
- **sistemul binar** este un sistem de numerație în baza 2, numărul de cifre utilizate este 2, adică 0 și 1. Cifrele respective se numesc **cifre binare** sau **biți**;
- **sistemul ternar** este un sistem de numerație în baza 3, numărul de cifre utilizate fiind 3, respectiv, 0, 1 și 2;
- **sistemul octal** este un sistem de numerație în baza 8, conținând 8 cifre: 0, 1, 2, ..., 7;
- **sistemul hexazecimal** este un sistem de numerație în baza 16 și conține 16 cifre: 0, 1, 2, ..., 9, *A* (zece), *B* (unsprezece), *C* (doisprezece), *D* (treisprezece), *E* (paisprezece), *F* (cincisprezece).

În *tabelul 3.1* sunt date reprezentările unora și acelorași numere în diferite baze.

Tabelul 3.1

Reprezentarea unor numere în diferite baze

Zecimal	Binar	Octal	Hexa-zecimal	Zecimal	Binar	Octal	Hexa-zecimal
0	0	0	0	7	111	7	7
1	1	1	1	8	1000	10	8
2	10	2	2	9	1001	11	9
3	11	3	3	10	1010	12	A
4	100	4	4	11	1011	13	B
5	101	5	5	12	1100	14	C
6	110	6	6	13	1101	15	D

Zecimal	Binar	Octal	Hexa-zecimal
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
...	...	...	...
30	11110	36	1E
...	...	...	...
40	101000	50	28
...	...	...	...

Zecimal	Binar	Octal	Hexa-zecimal
50	110010	62	32
...	...	...	...
60	111100	74	3C
...	...	...	...
70	1000110	106	46
...	...	...	...
80	1010000	120	50
...	...	...	...
90	1011010	132	5A
...	...	...	...
100	1100100	144	64
...	...	...	...

Regula de reprezentare a numerelor în sistemul zecimal rezultă din următorul exemplu:

$$(3856,43)_{10} = 3 \cdot 10^3 + 8 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0 + 4 \cdot 10^{-1} + 3 \cdot 10^{-2}.$$

Se observă că în această reprezentare semnificația (valoarea) fiecărei cifre depinde de poziția pe care o ocupă în număr. De exemplu, cifra 3 se întâlnește de 2 ori: prima dată are semnificația 3 000, iar a doua oară – semnificația 0,03.

Sistemele în care semnificația cifrelor depinde de poziția ocupată în cadrul numerelor se numesc sisteme de numerație poziționale.

Presupunem că numărul N are partea întreagă formată din $n+1$ cifre, iar partea fracționară – din m cifre:

$$N = c_n c_{n-1} \dots c_1 c_0, c_{-1} c_{-2} \dots c_{-m}.$$

Valoarea acestui număr se evaluează în funcție de baza sistemului:

$$(N)_b = c_n b^n + c_{n-1} b^{n-1} + \dots + c_1 b^1 + c_0 b^0 + c_{-1} b^{-1} + c_{-2} b^{-2} + \dots + c_{-m} b^{-m}.$$

Efectuând calculele respective, se va realiza **conversiunea** numărului $(N)_b$ din baza b în sistemul zecimal.

De exemplu,

$$(101,1)_{10} = 1 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0 + 1 \cdot 10^{-1} = 101,1;$$

$$(101,1)_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} = 5,5;$$

$$(101,1)_3 = 1 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 + 1 \cdot 3^{-1} = 10,333\dots;$$

$$(101,1)_8 = 1 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0 + 1 \cdot 8^{-1} = 65,125;$$

$$(101,1)_{16} = 1 \cdot 16^2 + 0 \cdot 16^1 + 1 \cdot 16^0 + 1 \cdot 16^{-1} = 257,0625.$$

Formal, sistemul zecimal nu prezintă niciun avantaj deosebit față de alte sisteme de numerație. Se presupune că acest sistem a fost adoptat încă din cele mai vechi timpuri datorită faptului că procesul de numărare a folosit ca instrumente inițiale degetele mâinilor.

Un calculator poate fi prevăzut să funcționeze în orice sistem de numerație. Pe parcursul dezvoltării tehnicii de calcul, s-a stabilit că cel mai avantajos este sistemul binar. Acest sistem a fost preferat din următoarele motive:

- simplitatea regulilor pentru operațiile aritmetice și logice;
- materializarea fizică a cifrelor în procesul prelucrării sau stocării numerelor se face mai ușor pentru două simboluri decât pentru zece: perforat–neperforat, contact închis–contact deschis, prezență sau absență de curent etc.;
- circuitele care trebuie să diferențieze numai între două stări sunt mai sigure în funcționare decât cele care trebuie să diferențieze între zece stări.

Menționăm că în procesul dezvoltării civilizației umane au fost create și **sisteme de numerație nepoziționale**. Drept exemplu poate servi **sistemul roman**, care utilizează cifrele *I, V, X, L, C, D, M*. Întrucât regulile de reprezentare a numerelor și de efectuare a operațiilor aritmetice sunt foarte complicate, sistemele nepoziționale au o utilizare foarte restrânsă.

Întrebări și exerciții

- ❶ Cum se definește un sistem de numerație?
- ❷ Care este deosebirea dintre sistemele de numerație poziționale și nepoziționale?
- ❸ Dați exemple de sisteme de numerație poziționale. Cum se definește baza sistemului de numerație?
- ❹ Evaluați numărul $(101,1)_b$ scris în următoarele sisteme de numerație:

$$b = 4, 5, 6, 7, 9, 11, 12, 13, 14 \text{ și } 15.$$

- ❺ Evaluați numerele ce urmează:

a) $(328)_9;$	i) $(1010,01)_3;$	q) $(341,02)_8;$
b) $(516)_7;$	j) $(201,12)_4;$	r) $(ABCD)_{16};$
c) $(1010,01)_2;$	k) $(341,02)_6;$	s) $(328)_{16};$
d) $(201,12)_3;$	l) $(1111)_{16};$	t) $(516)_{16};$
e) $(341,02)_5;$	m) $(328)_{11};$	u) $(1010,01)_{16};$
f) $(FFFF)_{16};$	n) $(516)_9;$	v) $(201,12)_{16};$
g) $(328)_{10};$	o) $(1010,01)_8;$	w) $(341,02)_{12};$
h) $(516)_8;$	p) $(201,12)_8;$	x) $(F001)_{16}.$

- ❻ Care factori au contribuit la utilizarea în tehnica de calcul a sistemului binar?
- ❼ Elaborați un program care evaluează numerele scrise în baza b , $b \leq 10$.
- ❽ Elaborați un program care evaluează numerele scrise în baza b , $10 \leq b \leq 36$.

- 9 CERCETEAZĂ! Explorând spațiul digital virtual, aflați ce sisteme de numerație utilizau marile civilizații antice. Cum au influențat aceste sisteme de numerație civilizațiile moderne?

3.2. Conversiunea numerelor dintr-un sistem în altul

Conversiunea numărului $(N)_b$ în echivalentul său zecimal se efectuează conform formulei din paragraful precedent:

$$(N)_b = c_n b^n + c_{n-1} b^{n-1} + \dots + c_1 b^1 + c_0 b^0 + c_{-1} b^{-1} + c_{-2} b^{-2} + \dots + c_{-m} b^{-m}.$$

Conversiunea numărului zecimal $(N)_{10}$ în echivalentul său în baza b se efectuează conform următoarelor reguli:

- se împarte la baza respectivă partea întreagă și câturile obținute după fiecare împărțire, până se obține câtul zero; rezultatul conversiunii părții întregi este constituit din resturile obținute, considerate în ordinea inversă de apariție;
- se înmulțește cu baza partea fracționară, apoi toate părțile fracționare obținute din produsul anterior, până când partea fracționară a unui produs este zero sau până la obținerea unui număr de cifre fracționare dorit; rezultatul conversiunii părții fracționare este constituit din părțile întregi ale produselor, considerate în ordinea apariției.

Să analizăm câteva exemple.

1) Să se transforme numărul zecimal 53,40625 în echivalentul său binar.

$$\begin{aligned} 53 : 2 &= 26 + \frac{1}{2}; \\ 26 : 2 &= 13 + \frac{0}{2}; \\ 13 : 2 &= 6 + \frac{1}{2}; \\ 6 : 2 &= 3 + \frac{0}{2}; \\ 3 : 2 &= 1 + \frac{1}{2}; \\ 1 : 2 &= 0 + \frac{1}{2}. \end{aligned}$$

Prin urmare, partea întreagă a numărului binar va fi 110101.

$$\begin{aligned} 0,40625 \times 2 &= 0,8125; \\ 0,8125 \times 2 &= 1,625; \\ 0,625 \times 2 &= 1,25; \\ 0,25 \times 2 &= 0,5; \\ 0,5 \times 2 &= 1,0. \end{aligned}$$

Partea fracționară a numărului binar va fi 01101. Prin urmare,

$$(53,40625)_{10} = (110101,01101)_2.$$

2) Să se transforme numărul 23,7 din sistemul zecimal în sistemul binar.

$$\begin{aligned} 23 : 2 &= 11 + \frac{1}{2}; \\ 11 : 2 &= 5 + \frac{1}{2}; \\ 5 : 2 &= 2 + \frac{1}{2}; \end{aligned}$$

$$2 : 2 = 1 + {}^0/2;$$

$$1 : 2 = 0 + {}^1/2;$$

$$0,7 \times 2 = 1,4;$$

$$0,4 \times 2 = 0,8;$$

$$0,8 \times 2 = 1,6;$$

$$0,6 \times 2 = 1,2;$$

$$0,2 \times 2 = 0,4;$$

$$0,4 \times 2 = 0,8;$$

...

Se observă că operația poate fi continuată la infinit, adică nu există o conversiune exactă a numărului analizat. Prin urmare,

$$(23,7)_{10} = (10111,101100\dots)_2.$$

3) Să se efectueze conversiunea numărului 1996,0625 din sistemul zecimal în sistemul octal.

$$1996 : 8 = 249 + {}^4/8;$$

$$249 : 8 = 31 + {}^1/8;$$

$$31 : 8 = 3 + {}^7/8;$$

$$3 : 8 = 0 + {}^3/8;$$

$$0,0625 \times 8 = 0,5;$$

$$0,5 \times 8 = 4.$$

Prin urmare,

$$(1996,0625)_{10} = (3714,04)_8.$$

4) Să se transforme numărul 2914,25 din sistemul zecimal în sistemul hexazecimal.

$$2914 : 16 = 182 + {}^2/16;$$

$$182 : 16 = 11 + {}^6/16;$$

$$11 : 16 = 0 + {}^{11}/16;$$

$$0,25 \times 16 = 4.$$

Prin urmare,

$$(2914,25)_{10} = (B62,4)_{16}.$$

Întrebări și exerciții

- 1 Cum se efectuează conversiunea unui număr dintr-un sistem cu baza b în sistemul zecimal?
- 2 Transformați în sistemul zecimal numerele ce urmează:

a) $(100001,01111)_2;$

b) $(328,678)_9;$

c) $(100,100)_{16};$

- | | | |
|----------------------|---------------------|---------------------|
| d) $(10,01)_{16};$ | g) $(1221,1112)_3;$ | j) $(4231,124)_5;$ |
| e) $(AAA,BBB)_{16};$ | h) $(1321,1312)_4;$ | k) $(50,505050)_6;$ |
| f) $(EE,00F)_{16};$ | i) $(124,521)_7;$ | l) $(7777,001)_8.$ |

❸ Cum se efectuează conversiunea unui număr zecimal în echivalentul său în baza b ?

❹ Transformați în sistemul binar numerele zecimale ce urmează:

- | | | |
|------------|------------|------------|
| a) 13,889; | e) 93,447; | i) 58,749; |
| b) 38,668; | f) 70,212; | j) 4,345; |
| c) 53,536; | g) 8,347; | k) 3,156; |
| d) 29,261; | h) 39,764; | l) 91,428. |

Verificați rezultatele efectuând conversiunea binar-zecimală.

❺ Transformați în sistemul octal numerele zecimale ce urmează:

- | | | |
|-------------|-------------|-------------|
| a) 358,932; | e) 886,526; | i) 795,128; |
| b) 479,093; | f) 971,258; | j) 680,895; |
| c) 591,241; | g) 515,914; | k) 256,453; |
| d) 649,113; | h) 347,607; | l) 838,261. |

Verificați rezultatele efectuând conversiunea octal-zecimală.

❻ Transformați în sistemul hexazecimal numerele zecimale ce urmează:

- | | | |
|--------------|--------------|--------------|
| a) 1424,699; | e) 5818,961; | i) 4985,995; |
| b) 3517,315; | f) 9336,491; | j) 9721,678; |
| c) 9607,201; | g) 3442,722; | k) 5292,837; |
| d) 8974,664; | h) 4521,449; | l) 2734,592. |

Verificați rezultatele efectuând conversiunea hexazecimal-zecimală.

❼ Elaborați un program care transformă numerele zecimale în echivalentele respective din sistemul de numerație în baza b , $b < 10$.

❽ Elaborați un program pentru transformarea numerelor zecimale în numere din sistemul în baza b , $10 < b \leq 36$.

3.3. Conversiunea din binar în octal, hexazecimal și invers

Întrucât $8 = 2^3$, **conversiunea binar-octală** și **octal-binară** se poate face direct. Orice cifră octală se reprezintă prin 3 cifre binare:

0 = 000;

1 = 001;

$$2 = 010;$$

$$5 = 101;$$

$$3 = 011;$$

$$6 = 110;$$

$$4 = 100;$$

$$7 = 111.$$

Dacă este dat un număr octal, pentru conversiunea lui în binar se va scrie fiecare cifră octală prin 3 cifre binare.

Exemple:

$$(247,315)_8 = (010\ 100\ 111, 011\ 001\ 101)_2;$$

$$(512,07)_8 = (101\ 001\ 010, 000\ 111)_2;$$

$$(3,146)_8 = (011, 001\ 100\ 110)_2.$$

Dacă este considerat un număr binar, pentru conversiunea lui în octal se vor grupa câte 3 cifre binare pornind de la poziția virgulei spre stânga pentru partea întreagă, respectiv dreapta pentru partea fracționară, aflând corespondentul în octal. Pentru completarea unui grup de trei cifre binare, zerourile din fața numărului, respectiv după ultima cifră a părții fracționare, nu modifică semnificația numărului.

Exemple:

$$(11,011101)_2 = (011,011\ 101)_2 = (3,35)_8;$$

$$(10,11011)_2 = (010,110\ 110)_2 = (2,66)_8;$$

$$(1001,01011)_2 = (001\ 001,010\ 110)_2 = (11,26)_8.$$

În mod similar se procedează și în cazul sistemului hexazecimal, baza căruia este $16 = 2^4$. Orice cifră hexazecimală se reprezintă prin 4 cifre binare:

$$0 = 0000;$$

$$8 = 1000;$$

$$1 = 0001;$$

$$9 = 1001;$$

$$2 = 0010;$$

$$A = 1010;$$

$$3 = 0011;$$

$$B = 1011;$$

$$4 = 0100;$$

$$C = 1100;$$

$$5 = 0101;$$

$$D = 1101;$$

$$6 = 0110;$$

$$E = 1110;$$

$$7 = 0111;$$

$$F = 1111.$$

Exemple:

$$(6AF3,B2)_{16} = (0110\ 1010\ 1111\ 0011, 1011\ 0010)_2;$$

$$(6F1,3CA)_{16} = (0110\ 1111\ 0001, 0011\ 1100\ 1010)_2;$$

$$(11,01101)_2 = (0011, 0110\ 1000)_2 = (3,68)_{16};$$

$$(10001,01011)_2 = (0001\ 0001, 0101\ 1000)_2 = (11,58)_{16}.$$

Întrebări și exerciții

❶ Cum se efectuează conversiunea octal-binară și binar-octală?

❷ Transformați în sistemul binar numerele octale ce urmează:

- | | | |
|------------|------------|------------|
| a) 15,006; | e) 21,626; | i) 42,322; |
| b) 13,06; | f) 6,3415; | j) 44,523; |
| c) 43,15; | g) 771,25; | k) 32,271; |
| d) 10,01; | h) 12,121; | l) 73,536. |

❸ Transformați în sistemul octal numerele binare ce urmează:

- | | | |
|-------------------|------------------|-------------------|
| a) 1,1; | e) 1101,1; | i) 10110,001011; |
| b) 101,10101; | f) 1,000001; | j) 11111,0010001; |
| c) 1111,000101; | g) 11110001,101; | k) 11001,00101; |
| d) 10101110,1001; | h) 0,00001; | l) 1011,0001011. |

❹ Elaborați un program pentru conversiunea binar-octală și octal-binară.

❺ Cum se efectuează conversiunea hexazecimal-binară și binar-hexazecimală?

❻ Transformați în sistemul binar numerele hexazecimale ce urmează:

- | | | |
|-------------|---------------|-------------|
| a) FFF,AAA; | e) 3,1AB; | i) C,DC1; |
| b) F,1A; | f) AAB,0000F; | j) 942,14A; |
| c) 1,009; | g) 81,91A; | k) CAA,B; |
| d) 0,00F; | h) 10,01; | l) DAD,ABA. |

❼ Transformați în sistemul hexazecimal numerele binare ce urmează:

- | | | |
|-----------------------|--------------------|----------------|
| a) 1001011101,01101; | e) 1011101,011; | i) 1011,0101; |
| b) 111,01; | f) 11,001011101; | j) 11001,0110; |
| c) 1,1; | g) 11110,01111; | k) 0,00001; |
| d) 10101110111,00101; | h) 111011,0010000; | l) 101,01011. |

❽ Elaborați un program pentru conversiunea binar-hexazecimală și hexazecimal-binară.

❾ Transformați în sistemul hexazecimal numerele octale ce urmează:

- | | | |
|-------------|-------------|-------------|
| a) 13,146; | e) 451,35; | i) 644,031; |
| b) 613,12; | f) 12,357; | j) 5,4312; |
| c) 541,723; | g) 53,627; | k) 675,542; |
| d) 104,527; | h) 572,004; | l) 372,271. |

10 Transformați în sistemul octal numerele hexazecimale ce urmează:

a) AA;

e) 15F,6A1;

i) BBB;

b) A2B,1F;

f) 3,281;

j) AF,31B;

c) F,3A;

g) 9,AF;

k) 2C,ACB;

d) FFFF;

h) 3,418F;

l) A1B,39E.

11 Elaborați un program pentru conversiunea octal-hexazecimală și hexazecimal-octală.

3.4. Operații aritmetice în binar

Operațiile aritmetice cu numerele binare sunt foarte simple. Regulile de operare în sistemul binar sunt prezentate în *tabelele 3.2, 3.3 și 3.4*.

Tabelul 3.2
Adunarea binară

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$

Tabelul 3.3
Scăderea binară

$0 - 0 = 0$
$1 - 0 = 1$
$1 - 1 = 0$
$10 - 1 = 1$

Tabelul 3.4
Înmulțirea binară

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

Exemple:

1) Se propune adunarea în binar a numerelor zecimale 29 și 43:

$$(29)_{10} = (11101)_2;$$

$$(43)_{10} = (101011)_2;$$

11101 +
101011
1001000

Verificare: $(1001000)_2 = (72)_{10}$, rezultatul este corect, întrucât $(29)_{10} + (43)_{10} = (72)_{10}$.

2) Se propune scăderea în binar a numărului zecimal 37 din numărul zecimal 46:

$$(37)_{10} = (100101)_2;$$

$$(46)_{10} = (101110)_2;$$

101110 _
100101
1001

Verificare: $(1001)_2 = (9)_{10}$, rezultatul este corect, întrucât $(46)_{10} - (37)_{10} = (9)_{10}$.

3) Se propune înmulțirea în binar a numerelor zecimale 3,25 și 7,125:

$$(3,25)_{10} = (11,01)_2;$$

$$(7,125)_{10} = (111,001)_2;$$

$$\begin{array}{r}
 11,01 \times \\
 111,001 \\
 \hline
 1101 \\
 0000 \\
 0000 \\
 1101 \\
 1101 \\
 1101 \\
 \hline
 10111,00101
 \end{array}$$

Verificare: $(10111,00101)_2 = (23,15625)_{10}$, rezultatul este corect, întrucât $(3,25)_{10} \times (7,125)_{10} = (23,15625)_{10}$.

4) Se propune împărțirea în binar a numărului zecimal 211 la numărul zecimal 3:

$$(211)_{10} = (11010011)_2;$$

$$(3)_{10} = (11)_2;$$

$$\begin{array}{r|l}
 11010011 & 11 \\
 \hline
 11 & 1000110 \\
 \hline
 00100 & \\
 11 & \\
 11 & \\
 11 & \\
 \hline
 01 &
 \end{array}$$

Deci $11010011 : 11 = 1000110$, rest 1.

Verificare: $(1000110)_2 = (70)_{10}$, rezultatul este corect, întrucât $(211)_{10} : (3)_{10} = (70)_{10} + (1)_{10}$.

Menționăm că atât la înmulțire, cât și la împărțire, fixarea virgulei care desparte partea întreagă de cea fracționară se face la fel ca și în sistemul de numerație zecimal.

Întrebări și exerciții

❶ Cum se execută operațiile aritmetice în sistemul binar?

❷ Calculați în sistemul binar:

a) $34 + 251;$

f) $1996 - 51;$

k) $0,5 \times 0,5;$

b) $68 - 7;$

g) $2015 + 1995;$

l) $1 : 0,5;$

c) $1512 + 620;$

h) $28,5 + 0,75;$

m) $40 : 0,125;$

d) $14 \times 8;$

i) $63,125 - 4,125;$

n) $32 : 2;$

e) $63 : 3;$

j) $3,0625 \times 2,125;$

o) $32 : 16;$

$$p) \quad 401 \times 8;$$

$$r) \quad 401 \times 4;$$

$$t) \quad 401 \times 2;$$

$$q) \quad 32 : 8;$$

$$s) \quad 32 : 4;$$

$$u) \quad 933 : 3.$$

Numerele analizate sunt scrise în sistemul zecimal.

- ③ Elaborați un program pentru adunarea și scăderea numerelor binare.
- ④ Elaborați un program pentru înmulțirea și împărțirea numerelor binare.

3.5. Reprezentarea numerelor naturale în calculator

Calculatoarele actuale utilizează sistemul de numerație binar. Reprezentarea numerelor naturale $N = \{0, 1, 2, \dots\}$ se realizează pe un număr fix de poziții binare, de regulă, 8, 16, 32 sau 64 (fig. 3.1).

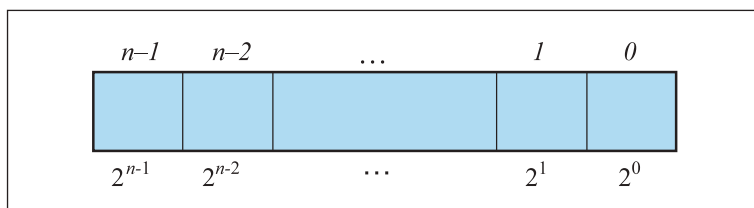


Fig. 3.1. Reprezentarea numerelor naturale pe n poziții binare

În pozițiile 0, 1, ..., $n-1$ sunt înscrise cifrele binare ale numărului natural reprezentat în sistemul de numerație binar. Alinierea numerelor binare se realizează la dreapta, eventualele zerouri nesemnificative sunt plasate în fața numărului binar.

Drept exemplu, în figura 3.2 este redată reprezentarea numărului natural

$$1039 = (10000001111)_2$$

pe un număr de 16 poziții binare.

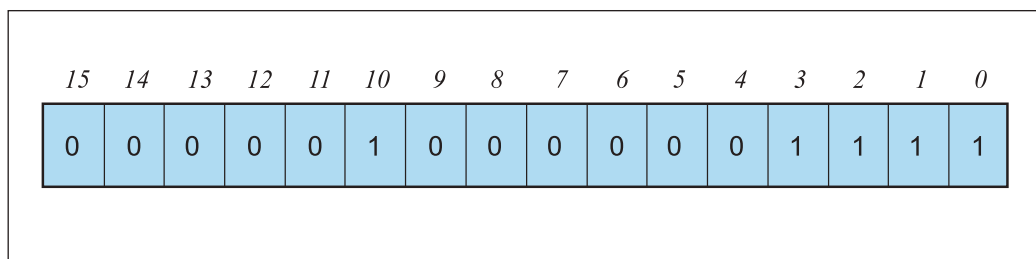


Fig. 3.2. Reprezentarea numărului natural 1039 pe 16 poziții binare

Numărul maxim ce poate fi reprezentat pe n poziții binare (fig. 3.3) este

$$1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + \dots + 1 \cdot 2^{n-1} = 2^n - 1.$$

Prin urmare, pe n poziții binare pot fi reprezentate numere naturale din intervalul $[0; 2^n - 1]$.

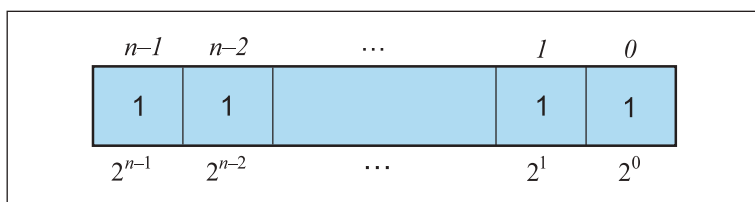


Fig. 3.3. Reprezentarea numărului natural maxim pe n poziții binare

Întrebări și exerciții

- ❶ Cum se reprezintă numerele naturale în calculator?
- ❷ Reprezentați numerele naturale 3, 112, 191, 204, 255 pe 8 poziții binare.
- ❸ Reprezentați numerele naturale 3, 255, 1 024, 2 048, 4 096, 65 535 pe 16 poziții binare.
- ❹ Calculați numerele naturale maxime ce pot fi reprezentate pe 4, 8, 12, 16, 24, 32 și 64 de poziții binare.
- ❺ CERCETEAZĂ! Aflați pe câte poziții binare sunt reprezentate numerele naturale în echipamentele digitale cu care lucrați.

3.6. Reprezentarea numerelor întregi

În calculator nu există posibilitatea introducerii directe a semnelor $+$ și $-$, atașate numerelor pozitive și negative. Din acest motiv, reprezentarea semnului numărului x se face cu ajutorul unei cifre binare, denumită **cifră-semn**, așezată în poziția $n-1$ (fig. 3.4):

$$S = \begin{cases} 0, & x > 0; \\ 1, & x < 0. \end{cases}$$

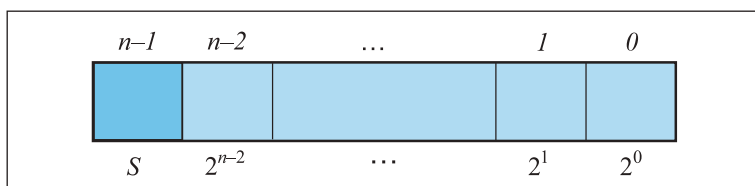


Fig. 3.4. Reprezentarea numerelor întregi pe n poziții binare

Numerele cu semn se reprezintă cel mai des nu în sistemul binar direct, ci într-un sistem binar codificat care oferă anumite avantaje în executarea operațiilor aritmetice cu numere algebrice. Din acest punct de vedere, se cunosc trei moduri de reprezentare, denumite **coduri binare pentru numere algebrice**.

Codul direct (codul mărime și semn). Scrierea unui număr în acest cod este foarte simplă: în cifra-semn se scrie 0, dacă numărul este pozitiv, și 1, dacă el este negativ; în partea de valoare se înscrie numărul în sistemul binar obișnuit.

Drept exemplu, în figura 3.5, p. 126 sunt redată reprezentările numerelor +52 și -52 pe 8 poziții binare.

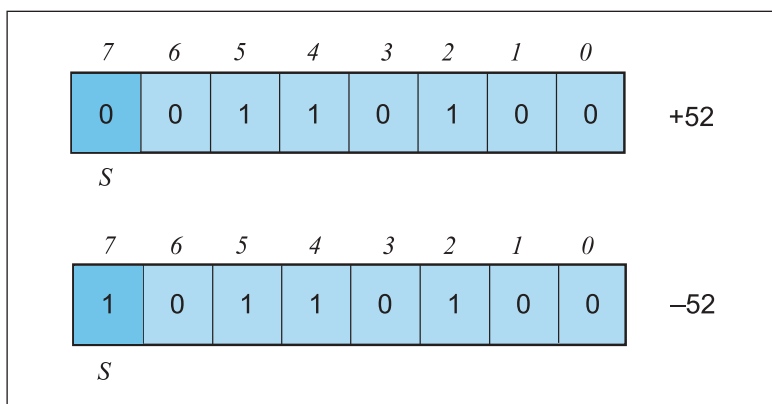


Fig. 3.5. Reprezentarea numerelor +52 și -52 în cod direct

În codul direct, pe n poziții binare se pot reprezenta numere întregi pozitive și negative N , astfel încât:

$$-(2^{n-1} - 1) \leq N \leq (2^{n-1} - 1).$$

Menționăm că în codul direct există două reprezentări binare pentru numărul 0: 00...0 și 10...0. Codul direct este rar utilizat în calculatoare, deoarece necesită algoritmi complicați de executare a operațiilor aritmetice și verificare a rezultatelor.

Codul invers. Pentru numerele pozitive scrierea în cod invers este identică cu cea din codul direct. Dacă numărul este negativ, el se scrie mai întâi ca și cum ar fi pozitiv, apoi se inversează fiecare cifră binară, adică 1 devine 0 și 0 devine 1.

În figura 3.6 sunt redată reprezentările numerelor +52 și -52 în cod invers pe 8 poziții binare.

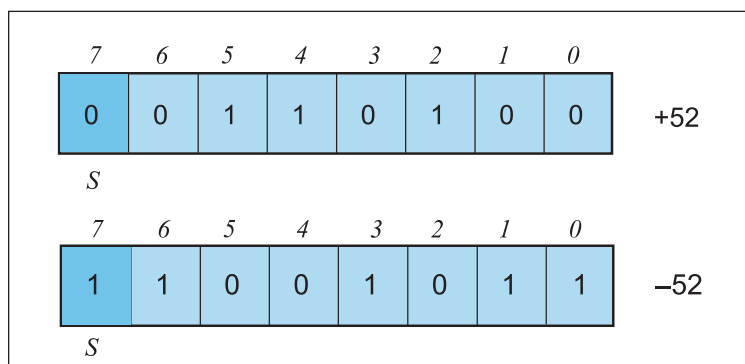


Fig. 3.6. Reprezentarea numerelor +52 și -52 în cod invers

Se poate ușor constata că intervalul numerelor întregi N care se pot reprezenta în cod invers este același ca și pentru reprezentarea în cod direct, iar numărul 0 are două reprezentări binare: 00...0 și 11...1.

Codul complementar. În acest cod numerele pozitive au aceeași reprezentare ca și în codul direct și codul invers. Dacă numărul este negativ, el se scrie mai întâi în codul invers, apoi se adună 1 la cifra cea mai puțin semnificativă (poziția binară 0).

Drept exemplu, în *figura 3.7* sunt redată reprezentările numerelor +52 și -52 în cod complementar.

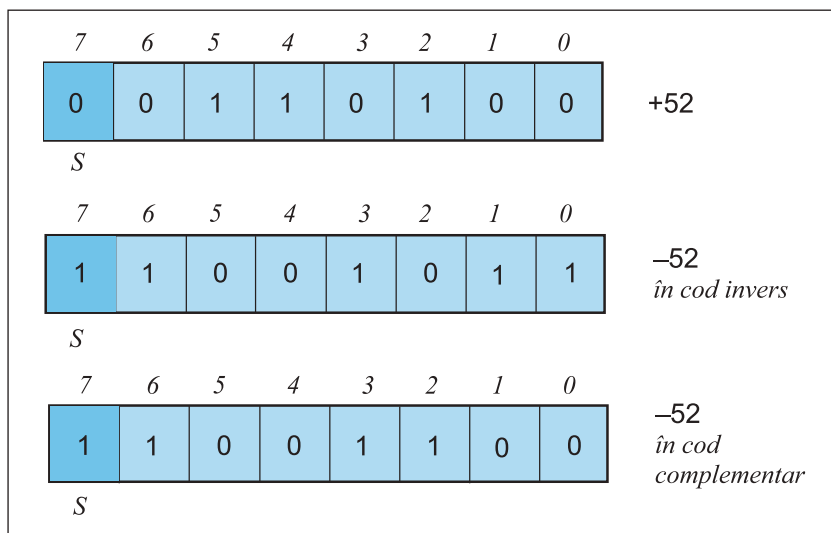


Fig. 3.7. Reprezentarea numerelor +52 și -52 în cod complementar

Codul complementar este utilizat în marea majoritate a calculatoarelor datorită facilităților pe care le oferă la efectuarea operațiilor aritmetice și ușurinței cu care se face verificarea rezultatelor. În acest cod pe n poziții binare pot fi reprezentate numere întregi din intervalul $[-2^{n-1}, 2^{n-1} - 1]$.

Întrebări și exerciții

- ❶ Cum pot fi reprezentate în calculator numerele întregi? Explicați cum se scriu numerele negative în cod direct, cod invers și cod complementar.
- ❷ Reprezentați în cod direct pe 8 poziții binare:

a) +12;	d) -12;	g) +21;
b) -21;	e) -64;	h) -68;
c) +68;	f) +105;	i) -112.
- ❸ Reprezentați în cod invers pe 8 poziții binare:

a) +10;	d) -10;	g) +65;
b) -65;	e) +101;	h) -101;
c) +112;	f) -112;	i) -105.
- ❹ Reprezentați în cod complementar pe 8 poziții binare:

a) +40;	c) +109;	e) -40;
b) +27;	d) -16;	f) -27;

g) -109;

i) +16;

k) +111;

h) -101;

j) +101;

l) -111.

- 5 Elaborati un program care afișează pe ecran reprezentările în cod direct, cod invers și cod complementar ale numerelor întregi introduse de la tastatură pentru $n = 8, 16$ și 32 .
- 6 Cum se reprezintă numărul 0 în cod direct, cod invers și cod complementar? Câte reprezentări are numărul 0 în codurile examinate?
- 7 Determinați numărul maxim ce poate fi reprezentat pe n poziții binare în cod direct, cod invers și cod complementar.
- 8 CERCETEAZĂ! Aflați pe câte poziții și ce coduri binare pentru numere algebrice sunt utilizate pentru reprezentare numerele întregi în echipamentele digitale cu care lucrați.

3.7. Reprezentarea numerelor reale

Numerele reale se reprezintă în calculator sub formă fracționară prin intermediul reprezentării în virgulă fixă sau în virgulă mobilă (virgulă flotantă).

Reprezentarea în virgulă fixă. În acest caz se consideră că toate numerele au virgula plasată în aceeași poziție, chiar dacă acest lucru nu corespunde formei externe de reprezentare. Procesul de translatore din forma externă în forma internă și invers se realizează cu ajutorul unor coeficienți de scalare aleși în mod corespunzător de programator.

De obicei, se consideră că virgula este plasată imediat după poziția cifrei-semn, caz în care numerele sunt fracții pure (fig. 3.8).

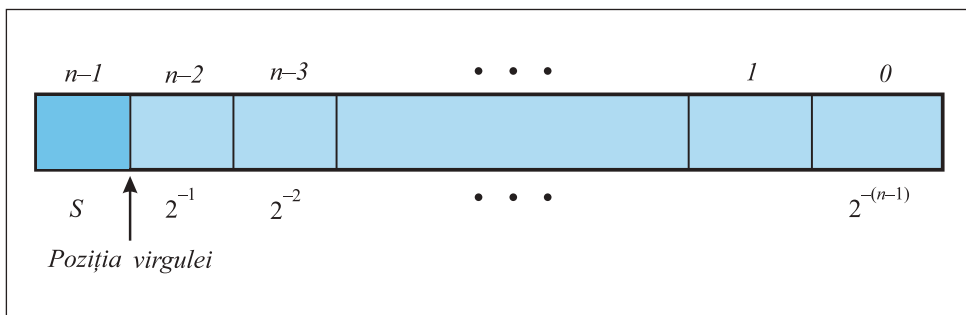


Fig. 3.8. Reprezentarea numerelor reale în virgulă fixă

Virgula însăși nu este materializată fizic în calculator. Din figura 3.8 rezultă că pe n poziții binare pot fi reprezentate numere reale, valoarea absolută a cărora este

$$0,00\dots0 \leq |x| \leq 0,11\dots1$$

sau, în sistemul zecimal,

$$0 \leq |x| \leq 1 - 2^{-(n-1)}.$$

Ca și în cazul numerelor întregi, numerele subunitare cu semn pot fi reprezentate, cu unele modificări, în cod direct, cod invers sau cod complementar.

Drept exemplu, în *figura 3.9* este redată reprezentarea numerelor

$$+0,9375 = +15/16 = (0,1111)_2,$$

$$-0,9375 = -15/16 = (-0,1111)_2$$

în virgulă fixă pe 8 poziții binare în cod direct.

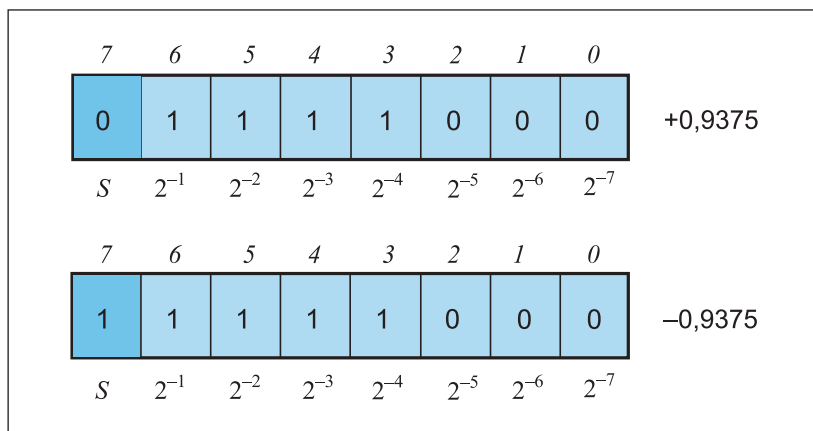


Fig. 3.9. Reprezentarea numerelor +0,9375 și -0,9375 în virgulă fixă în cod direct

Avantajul principal al reprezentării în virgulă fixă constă în faptul că operațiile aritmetice cu numere reale pot fi efectuate de dispozitivul aritmetic destinat operării cu numere întregi.

Reprezentarea în virgulă mobilă. Operațiile în virgulă fixă sunt comode pentru circuitele omogene de date, când toate numerele reale sunt subunitare. Însă această reprezentare este inefficientă în calculele științifice, unde se lucrează simultan cu numere foarte mari și numere foarte mici al căror ordin de mărime adesea este imprevizibil. Pentru astfel de probleme se utilizează reprezentarea în virgulă mobilă.

Numerele reprezentate în virgulă mobilă pot fi numere întregi sau fracționare a căror valoare este dată de relația:

$$x = M \times b^E,$$

unde b este valoarea bazei, M este un număr subunitar numit **mantisă**, iar E este un **exponent**. În calculatoarele actuale se utilizează $b = 2$ sau 16.

Numărul reprezentat în virgulă mobilă este **normalizat** dacă prima cifră după virgulă a mantisei este diferită de zero.

Exemple:

1) Numărul 23 se va exprima în virgulă mobilă astfel:

$$23 = (10111)_2 = 0,10111 \times 2^5,$$

unde $M = 0,10111$; $b = 2$; $E = 5$.

2) Numărul 4,9375 se scrie în virgulă mobilă în felul următor:

$$4,9375 = (100,1111)_2 = 0,1001111 \times 2^3,$$

$$M = 0,1001111; b = 2; E = 3.$$

3) Numărul $-0,375$ se scrie în virgulă mobilă după cum urmează:

$$-0,375 = (-0,011)_2 = -0,11 \times 2^{-1},$$

$$M = -0,11; b = 2; E = -1.$$

Se observă că poziția reală a virgulei în cadrul numărului depinde de valoarea exponentului, adică virgula este mobilă (flotantă).

Există mai multe variante de reprezentare a mantisei și a exponentului pe n poziții binare. În figura 3.10 este redată reprezentarea în virgulă mobilă în **formatul exponent-mantisă**.

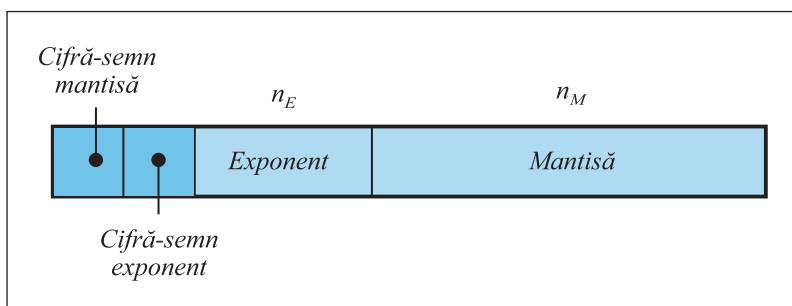


Fig. 3.10. Reprezentarea în virgulă mobilă, formatul exponent-mantisă

Numărul pozițiilor binare n_E alocate exponentului determină domeniul de mărime al numerelor care pot fi reprezentate, în timp ce numărul biților pentru mantisă n_M determină precizia de exprimare a numărului. În calculatoarele actuale $n_E = 6 \dots 15$ și $n_M = 24 \dots 64$, fapt ce permite reprezentarea numerelor într-un domeniu foarte larg al ordinului de mărime.

În figura 3.11 este redată reprezentarea numerelor în formatul **caracteristică-mantisă**.

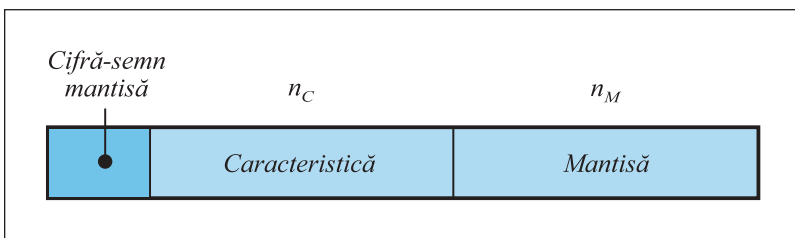


Fig. 3.11. Reprezentarea în virgulă mobilă, formatul caracteristică-mantisă

Caracteristica C constituie o formă de exprimare a exponentului E și se determină din relația

$$C = E + K.$$

În calculatoarele personale

$$K = 2^{n_c - 1} - 1,$$

unde n_c este numărul de poziții binare alocate caracteristicii. În această reprezentare caracteristica, spre deosebire de exponent, nu poate lua valori negative, fapt ce simplifică structura dispozitivelor aritmetice.

În particular, pentru $n = 8$, caracteristica este calculată astfel:

$$C = E + 127,$$

indicând practic semnul exponentului:

- dacă $C \geq 127$, atunci $E \geq 0$;
- dacă $C < 127$, atunci $E < 0$.

În *tabelul 3.5* sunt prezentate formatele caracteristică-mantisă, utilizate în majoritatea calculatoarelor personale.

Tabelul 3.5

Formatele caracteristică-mantisă în calculatoarele personale

Denumirea formatului	n	n_c	n_M	Precizia mantisei, cifre zecimale	Domeniul de mărime al numerelor
Simplă precizie	32	8	23	6 sau 7	$10^{-37} \dots 10^{38}$
Dublă precizie	64	11	52	15 sau 16	$10^{-307} \dots 10^{308}$
Precizie extinsă	80	15	64	19	$10^{-4932} \dots 10^{4932}$

Întrucât, conform condiției de normalizare, prima cifră a mantisei întotdeauna este cifra 1, acest bit poate să se reprezinte (să ocupe o poziție) sau nu în memoria calculatorului. În cazul în care nu se reprezintă, acest bit se numește **bit ascuns**. Accentuăm că tehnica bitului ascuns se referă doar la reprezentarea numerelor în memoria calculatorului, nu și la operațiile efectuate de dispozitivul aritmetic.

Întrebări și exerciții

- ❶ Care este deosebirea dintre reprezentarea în virgulă fixă și reprezentarea în virgulă mobilă?
- ❷ Reprezentați în virgulă fixă pe 8 poziții binare în cod direct:

a) +0,875;	e) -0,875;	i) +0,125;
b) -0,125;	f) +0,3;	j) -0,3;
c) +0,4;	g) -0,4;	k) +0,15625;
d) -0,15625;	h) +0,21875;	l) -0,21875.
- ❸ Care sunt avantajele și dezavantajele reprezentării în virgulă fixă?
- ❹ Cum se reprezintă numerele reale în virgulă mobilă? Care numere reprezentate în virgulă mobilă respectă condiția de normalizare?
- ❺ Cum se efectuează normalizarea numerelor reprezentate în virgulă mobilă?

⑥ Exprimați în virgulă mobilă următoarele numere:

a) +1,5;

e) -1,5;

i) -0,0625;

b) +0,0625;

f) +3,25;

j) +2,7;

c) -3,25;

g) -32,87;

k) -2,7;

d) -6,25;

h) +6,25;

l) -0,147.

- ⑦ De ce depinde precizia și domeniul de mărime ale numerelor reprezentate în virgulă mobilă?
- ⑧ Care este deosebirea dintre formatele exponent-mantisă și caracteristică-mantisă? Cum se calculează caracteristicile numerelor reprezentate în virgulă mobilă?
- ⑨ Calculați caracteristicile numerelor din exercițiul 6. Se consideră că $n_c = 8$.
- ⑩ Explicați termenul „bit ascuns”. Care sunt avantajele acestei reprezentări?
- ⑪ CERCETEAZĂ! Aflați cum se reprezintă numerele naturale, întregi și reale în calculatorul și echipamentele digitale la/cu care lucrați dvs.? Determinați numărul de poziții binare alocate fiecărei reprezentări.

4.1. Variabile și expresii logice

Algebra booleană sau **algebra logică** este un compartiment al matematicii în care legile gândirii – obiectul de studiu al logicii clasice – sunt studiate cu ajutorul metodelor simbolice. Denumirea aceasta a fost dată în onoarea matematicianului englez George Boole, care în lucrarea *The Laws of Thought* („Legile gândirii”), publicată în 1853, a pus bazele acestei algebre. Mulți ani algebra booleană a fost considerată drept o simplă curiozitate matematică, fără a i se găsi o utilizare într-un anumit domeniu al științelor aplicate. Ea a revenit în actualitate odată cu apariția centralelor telefonice automate și a calculatoarelor numerice.

Din punct de vedere formal, algebra booleană poate fi definită printr-o mulțime a elementelor $\{0, 1\}$, o mulțime a operatorilor elementari $\{\neg, \&, \vee\}$ și printr-un set de postulate. Prin urmare, în algebra booleană orice variabilă nu poate avea decât una dintre cele două valori posibile, notate simbolic prin 0 și 1, alte valori neavând nicio semnificație.

Variabilele algebrei booleene se notează prin $x, y, z, x_1, x_2, \dots$ cu sau fără indicii, iar elementele 0 și 1 se numesc **constante logice**.

Operatorii elementari ai algebrei booleene au următoarele denumiri:

- $\neg$ – negația (inversia, operația logică *NU*);
- $\&$ – conjuncția (produsul logic, operația logică *ȘI*);
- $\vee$ – disjuncția (suma logică, operația logică *SAU*).

Operatorii elementari se definesc cu ajutorul unor tabele speciale, numite **tabele de adevăr**.

Tabelul de adevăr este un tabel care include toate combinațiile posibile ale valorilor variabilelor față de care este definit operatorul și rezultatul operației respective.

În figura 4.1 sunt prezentate tabelele de adevăr ale negației, conjuncției și disjuncției.

x	$\bar{x}$
0	1
1	0

x	y	$x \& y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 4.1. Tabelele de adevăr ale operatorilor elementari

Menționăm că în algebra booleană negația se indică printr-o linie orizontală amplasată deasupra variabilei respective.

Întrucât variabila x poate avea numai valoarea 0 sau 1, tabelul de adevăr al negației conține două rânduri. În cazurile conjuncției și disjuncției tabelele de adevăr conțin patru rânduri, câte un rând pentru fiecare din combinațiile posibile 00, 01, 10 și 11 ale variabilelor x și y .

Variabilele și constantele logice, reunite cu ajutorul operatorilor $\neg$, $\&$ și $\vee$ formează **expresii logice**, de exemplu:

$$1) \ x \& y \vee z;$$

$$4) \ 1 \& x \vee y;$$

$$2) \ x \vee y \& z;$$

$$5) \ 0 \vee x \vee y.$$

$$3) \ \bar{x} \& y \vee z;$$

$$6) \ 1 \vee 0.$$

Valorile expresiilor logice pot fi calculate utilizând tabelele de adevăr ale operatorilor elementari. Pentru evaluarea expresiilor se stabilește următoarea **prioritate a operațiilor logice**:

- 1) negația;
- 2) conjuncția;
- 3) disjuncția.

De exemplu, în expresia

$$x \vee y \& z$$

mai întâi se execută operația $\&$, iar după aceea operația $\vee$. În expresia

$$x \& y \vee \bar{z}$$

se execută negația, în continuare conjuncția și, în sfârșit, disjuncția.

Ordinea executării operațiilor logice poate fi schimbată cu ajutorul parantezelor „(” și „)”. Evident, în primul rând, se execută operațiile dintre paranteze.

De exemplu, în cazul expresiei logice

$$x \vee y \& \bar{x} \vee z$$

se execută negația, conjuncția și disjuncțiile respective, iar în cazul expresiei

$$(x \vee y) \& (\bar{x} \vee z)$$

după negație se vor executa disjuncțiile și, pe urmă, conjuncția.

Pentru a sistematiza calculele, evaluarea expresiilor logice se efectuează în tabele speciale, numite **tabele de adevăr ale expresiilor logice**.

Tabelul de adevăr al expresiei logice include toate combinațiile posibile ale valorilor variabilelor din expresia examinată și rezultatele operațiilor logice în ordinea calculării lor.

De exemplu, în figura 4.2 este reprezentat tabelul de adevăr al expresiei,

$$\bar{x} \& y \vee z,$$

iar în figura 4.3 – tabelul de adevăr al expresiei

$$\overline{x \& y \vee z}.$$

x	y	z	$\bar{x}$	$\bar{x} \& y$	$\bar{x} \& y \vee z$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	0	0	0
1	1	1	0	0	1

Fig. 4.2. Tabelul de adevăr al expresiei logice $\bar{x} \& y \vee z$

x	y	z	$x \& y$	$x \& y \vee z$	$\overline{x \& y \vee z}$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	1	0

Fig. 4.3. Tabelul de adevăr al expresiei logice $\overline{x \& y \vee z}$

Menționăm că, pentru a simplifica notațiile, se admite ca simbolul operatorului $\&$ să fie omis din expresiile logice. De exemplu, expresiile logice

$$\bar{x} \& y \vee z,$$

$$(x \vee y) \& (\bar{x} \vee z)$$

pot fi scrise și în forma:

$$\bar{x}y \vee z,$$

$$(x \vee y)(\bar{x} \vee z).$$

Întrebări și exerciții

- 1 Ce valori poate avea o variabilă logică?
- 2 Care sunt operatorii elementari ai algebrei booleene și cum se definesc ei?
- 3 Memorizați tabelele de adevăr ale negației, conjuncției și disjuncției.
- 4 Cum se formează expresiile logice? Care este prioritatea operațiilor logice?
- 5 Explicați rolul parantezelor din componența expresiilor logice.

⑥ Alcătuiți tabelele de adevăr ale următoarelor expresii logice:

- | | |
|----------------------------|------------------------------|
| a) $\bar{x}y$; | h) $\bar{x} \vee \bar{y}$; |
| b) $\bar{x} \vee y$; | i) xyz ; |
| c) $\overline{x \vee y}$; | j) $x \vee y \vee z$; |
| d) $\bar{\bar{x}}$; | k) $xy \vee z$; |
| e) $\bar{x}x$; | l) $x(y \vee z)$; |
| f) $\overline{xy}$; | m) $\bar{x} \vee y \vee z$; |
| g) $\bar{x} \vee x$; | n) $\bar{x}(y \vee z)$. |

⑦ Elaborați un program care introduce valorile logice ale variabilelor x , y , z și afișează valorile uneia dintre expresiile ce urmează:

- | | |
|-------------------------------|-----------------------------------|
| a) $\bar{x}$; | f) $x \vee \bar{y} \vee z$; |
| b) xy ; | g) $xy \vee z$; |
| c) $x \vee y$; | h) $\overline{x \vee y \vee z}$; |
| d) $\overline{(x \vee y)}z$; | i) $x \vee xy$; |
| e) $\overline{xy} \vee z$; | j) $\bar{x} \vee y \vee z$. |

⑧ Elaborați un program care afișează tabelele de adevăr ale conjuncției și disjuncției.

⑨ Elaborați un algoritm repetitiv care formează toate combinațiile posibile ale valorilor variabilelor x , y și z . Afișați combinațiile respective pe ecran.

⑩ Pentru fiecare dintre expresiile logice ce urmează elaborați câte un program care alcătuieste tabelul de adevăr respectiv:

- | | |
|----------------------------|-----------------------------------|
| a) $x \vee \bar{x}$; | f) $\bar{x}y$; |
| b) $x\bar{x}$; | g) $x \vee y \vee z$; |
| c) $\overline{x \vee y}$; | h) xyz ; |
| d) $\overline{xy}$; | i) $(x \vee y)(\bar{x} \vee y)$; |
| e) $\bar{x} \vee y$; | j) $(x \vee y)(x \vee \bar{y})$. |

⑪ CREEAZĂ! Colectați informațiile necesare și scrieți un eseu despre viața, studiile și cariera profesională a matematicianului englez *George Boole*.

⑫ DESCOPERĂ! Spre deosebire de logica binară, adică logica ce operează doar cu două valori posibile, există și extensiuni ale acesteia, care operează cu trei și chiar cu mai multe valori. Explorând diverse surse de informații, aflați aceste extensiuni și domeniile în care ele sunt utilizate.

4.2. Funcții logice

Noțiunea de **funcție logică** sau **funcție booleană** se definește în același mod ca și în cazul algebrei clasice.

Vom nota prin $x_1, x_2, \dots, x_n$ un grup arbitrar de variabile booleene, unde $n = 1, 2, 3, \dots$. Întrucât fiecare variabilă booleană poate avea numai valorile 0 sau 1, numărul tuturor combinațiilor posibile ale valorilor variabilelor $x_1, x_2, \dots, x_n$ este de 2^n .

Firește, pentru $n = 1$ avem 2 combinații (0 și 1); pentru $n = 2$ sunt $2^2 = 4$ combinații (00, 01, 10 și 11); pentru $n = 3$ există $2^3 = 8$ combinații (000, 001, 010, 011, 100, 101, 110 și 111) etc.

Funcția logică de n variabile $y = f(x_1, x_2, \dots, x_n)$ este o aplicație care pune în corespondență fiecărei combinații de valori ale variabilelor $x_1, x_2, \dots, x_n$ valoarea 0 sau 1 a variabilei y .

Variabilele $x_1, x_2, \dots, x_n$ se numesc **variabile independente** sau **argumente**, iar variabila y – **variabilă dependentă** sau **funcție** de argumente $x_1, x_2, \dots, x_n$.

Prin urmare, **domeniul de definiție** al funcției $y = f(x_1, x_2, \dots, x_n)$ este mulțimea tuturor combinațiilor posibile

0 0 ... 0
0 0 ... 1
...
1 1 ... 1

ale valorilor argumentelor $x_1, x_2, \dots, x_n$, în total 2^n combinații, iar **domeniul valorilor** funcției logice este mulțimea $\{0, 1\}$.

Ca și în cazul algebrei clasice, funcțiile logice pot fi definite prin **tabele**, **formule** și **metode grafice**.

Tabelul de adevăr al funcției logice $y = f(x_1, x_2, \dots, x_n)$ este un tabel care include toate combinațiile posibile ale valorilor argumentelor $x_1, x_2, \dots, x_n$ și valorile corespunzătoare ale variabilei dependente y .

De exemplu, în figura 4.4 este prezentat tabelul de adevăr al unei funcții logice de 3 variabile.

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Fig. 4.4. Tabelul de adevăr al unei funcții logice de 3 variabile

Tabelul include $2^3 = 8$ rânduri și are 2 coloane: prima pentru combinațiile posibile ale valorilor argumentelor x_1, x_2, x_3 și a doua pentru valorile corespunzătoare ale variabilei dependente y . Conform tabelului analizat, combinației $x_1 = 0, x_2 = 0, x_3 = 0$ îi corespunde valoarea $y = f(0,0,0) = 1$; combinației $x_1 = 0, x_2 = 0, x_3 = 1$ – valoarea $y = f(0,0,1) = 0$ etc.

Definirea funcției logice prin formule se face atribuind variabilei dependente y valorile expresiilor logice ce conțin argumentele $x_1, x_2, \dots, x_n$.

De exemplu,

$$1) \quad y = x;$$

$$3) \quad y = \bar{x}_1 x_2 \vee x_3;$$

$$2) \quad y = x_1 x_2;$$

$$4) \quad y = x_1 x_2 \vee x_3.$$

Natural, cunoscând formula unei funcții logice, poate fi calculat tabelul ei de adevăr. De exemplu, tabelul de adevăr al funcției

$$y = \overline{x_1 x_2 \vee x_3}$$

va fi cel prezentat în *figura 4.4*. Ca să ne convingem de acest fapt, e suficient să calculăm tabelul de adevăr al expresiei logice

$$\overline{x_1 x_2 \vee x_3}$$

prezentat, prin alte notații, în *figura 4.3* din paragraful 4.1.

Există mai multe **metode grafice de definire a funcțiilor logice**. Aceste metode se bazează pe diagramele utilizate în teoria mulțimilor și se studiază în cursurile avansate de informatică.

Întrebări și exerciții

- ① Formulați definiția noțiunii *funcție logică*.
- ② Care sunt domeniul de definiție și domeniul de valori ale unei funcții logice?
- ③ Prin ce metode poate fi definită o funcție logică de n variabile?
- ④ Cum se alcătuiește tabelul de adevăr al unei funcții de n variabile? Câte rânduri conține acest tabel?
- ⑤ Cum poate fi alcătuit tabelul de adevăr al unei funcții logice atunci când se cunoaște formula ei?
- ⑥ Funcția logică de 3 variabile $y = f(x_1, x_2, x_3)$ este definită prin tabelul de adevăr din *figura 4.4*. Numiți combinațiile valorilor argumentelor x_1, x_2, x_3 pentru care funcția dată are valoarea $y = 1$. Numiți combinațiile respective pentru valoarea funcției $y = 0$.
- ⑦ Alcătuiți tabelele de adevăr ale următoarelor funcții logice:

$$a) \quad y = x;$$

$$f) \quad y = \overline{x_1 \vee x_2};$$

$$b) \quad y = \bar{x};$$

$$g) \quad y = \overline{x_1 x_2 x_3};$$

$$c) \quad y = x_1 x_2;$$

$$h) \quad y = x_1 (x_2 \vee \bar{x}_3);$$

$$d) \quad y = x_1 \vee x_2;$$

$$i) \quad y = \bar{x}_1 \vee x_2 x_3;$$

$$e) \quad y = \overline{x_1 x_2};$$

$$j) \quad y = x_1 \vee \overline{x_2 x_3}.$$

- 8 Elaborati un program care introduce valorile logice ale variabilelor x_1, x_2, x_3, x_4 și afișează valorile uneia dintre funcțiile ce urmează:

- | | |
|---|--|
| a) $y = x_1 x_2 \vee x_3 x_4;$ | g) $y = x_1 \bar{x}_2 \vee x_3 \bar{x}_4;$ |
| b) $y = (x_1 \vee x_2)(x_3 \vee x_4);$ | h) $y = \bar{x}_1 x_2 \vee \bar{x}_3 x_4;$ |
| c) $y = \overline{x_1 x_2 x_3 x_4};$ | i) $y = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4;$ |
| d) $y = \overline{x_1 \vee x_2 \vee x_3 \vee x_4};$ | j) $y = x_1 x_2 x_3 x_4 \vee \bar{x}_2 x_3 x_4;$ |
| e) $y = \overline{x_1 x_2 \vee x_3 x_4};$ | k) $y = x_1 \vee x_2 x_3 \vee x_4;$ |
| f) $y = \overline{(x_1 \vee x_2)(x_3 \vee x_4)};$ | l) $y = x_1 \vee x_2 \vee x_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4.$ |

- 9 Elaborati un program care alcătuiește tabelul de adevăr al uneia dintre funcțiile ce urmează:

- | | |
|------------------------------------|---|
| a) $y = x;$ | h) $y = \bar{x}_1 \bar{x}_2;$ |
| b) $y = \bar{x};$ | i) $y = \bar{x}_1 x_2 x_3;$ |
| c) $y = x_1 x_2;$ | j) $y = \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3;$ |
| d) $y = x_1 \vee x_2;$ | k) $y = \overline{x_1 \vee x_2 \vee x_3};$ |
| e) $y = \overline{x_1 x_2};$ | l) $y = \bar{x}_1 \bar{x}_2 \bar{x}_3;$ |
| f) $y = \bar{x}_1 \vee \bar{x}_2;$ | m) $y = x_1 \vee x_2 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4;$ |
| g) $y = \overline{x_1 \vee x_2};$ | n) $y = x_1(x_2 \vee \bar{x}_3 \vee \bar{x}_4).$ |

4.3. Funcții logice frecvent utilizate

Să admitem n variabile independente $x_1, x_2, \dots, x_n$. Apare întrebarea, câte funcții logice de n variabile există în algebra booleană? **Numărul funcțiilor logice posibile** poate fi determinat prin următoarele raționamente.

Întrucât orice funcție logică poate fi definită cu ajutorul tabelului de adevăr, numărul funcțiilor posibile de n variabile coincide cu numărul tabelelor distincte de adevăr.

Pentru a defini o funcție logică, în coloana y a tabelului de adevăr se indică valorile funcției – 0 sau 1 pentru fiecare dintre cele 2^n combinații ale valorilor argumentelor. Întrucât tabelul de adevăr are 2^n rânduri, există

$$m = 2^{2^n}$$

funcții logice de n variabile. Funcțiile respective se notează prin $y_j, j = 0, 1, \dots, m - 1$.

De exemplu, în cazul în care $n = 1$, există $m = 2^{2^1} = 2^2 = 4$ funcții logice, reprezentate în figura 4.5.

x	y_0	y_1	y_2	y_3
0	0	1	0	1
1	0	0	1	1

Fig. 4.5. Funcții logice de o singură variabilă

Evident,

$$y_0 = f(x) = 0;$$

$$y_1 = f(x) = \bar{x};$$

$$y_2 = f(x) = x;$$

$$y_3 = f(x) = 1.$$

Funcțiile y_0 și y_3 se numesc **funcția constanta 0** și, respectiv, **constantă 1**. Funcția y_1 este **funcția logică NU** sau **negația**, iar funcția y_2 se numește **funcția de repetare**.

Pentru $n = 2$ există

$$m = 2^{2^2} = 2^4 = 16$$

funcții logice, reprezentate în *figura 4.6*.

$x_1 x_2$	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}
0 0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0 1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Fig. 4.6. Funcții logice de două variabile

Funcțiile y_0 și y_{15} sunt funcțiile deja cunoscute, constanta 0 și, respectiv, constanta 1:

$$y_0 = f(x_1, x_2) = 0;$$

$$y_{15} = f(x_1, x_2) = 1.$$

Funcția y_8 poate fi scrisă în forma:

$$y_8 = f(x_1, x_2) = x_1 x_2.$$

În mod firesc, funcția y_8 va avea denumirea **funcția logică ȘI** sau **conjuncția**.

Funcția y_{14} poate fi scrisă în forma:

$$y_{14} = f(x_1, x_2) = \overline{x_1 \vee x_2}.$$

Prin urmare, funcția y_{14} va avea denumirea **funcția logică SAU** sau **disjuncția**.

Funcțiile logice NU, ȘI, SAU, induse de operatorii elementari, respectiv, $\bar{}$, $\&$, $\vee$ se numesc funcții logice elementare.

Din *figura 4.6* se observă că

$$y_1 = x_1 \vee x_2.$$

Funcția dată se numește **funcția logică SAU-NU**.

În mod similar, funcția y_7 poate fi exprimată în forma

$$y_7 = f(x_1, x_2) = \overline{x_1 x_2}.$$

Funcția în cauză se numește **funcția logică ȘI-NU**.

Funcția

$$y_9 = f(x_1, x_2) = \overline{x_1} \overline{x_2} \vee x_1 x_2$$

are valoarea 1 numai când $x_1 = x_2 = 0$ sau $x_1 = x_2 = 1$. Această funcție se numește **funcția logică COINCIDENȚĂ** sau **echivalență** și se notează prin simbolul „ $\equiv$ ”.

Analizând tabelul din *figura 4.6*, mai observăm că

$$\begin{aligned} y_3 &= f(x_1, x_2) = \bar{x}_1 \text{ (negația variabilei } x_1); \\ y_{12} &= f(x_1, x_2) = x_1 \text{ (repetarea variabilei } x_1); \\ y_5 &= f(x_1, x_2) = \bar{x}_2 \text{ (negația variabilei } x_2); \\ y_{10} &= f(x_1, x_2) = x_2 \text{ (repetarea variabilei } x_2). \end{aligned}$$

Pentru a fi mai ușor memorizate, în *figura 4.7* sunt prezentate tabelele de adevăr ale funcțiilor logice *NU*, *ȘI*, *SAU*, *ȘI-NU*, *SAU-NU* și *COINCIDENTĂ*.

<i>NU</i>		<i>ȘI</i>		<i>SAU</i>	
x	$\bar{x}$	x_1	x_2	$x_1 x_2$	$x_1 \vee x_2$
0	1	0	0	0	0
1	0	0	1	0	1
		1	0	0	1
		1	1	1	1

<i>ȘI-NU</i>		<i>SAU-NU</i>		<i>COINCIDENTĂ</i>	
x_1	x_2	$\overline{x_1 x_2}$	x_1	x_2	$\overline{x_1 \vee x_2}$
0	0	1	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	1

Fig. 4.7. Funcțiile logice frecvent utilizate

În mod similar pot fi studiate funcțiile logice de 3 variabile, numărul cărora este de $m = 2^3 = 2^8 = 256$; funcțiile de 4 variabile, numărul cărora este de $m = 2^4 = 2^{16} = 65\,536$ etc. Se observă că, deși este finit, numărul funcțiilor booleene posibile este enorm. S-a demonstrat însă că **orice funcție logică de n variabile, $n \geq 2$, poate fi exprimată printr-o formulă care include numai operatorii elementari $\bar{\cdot}$, $\&$, $\vee$** . Această proprietate facilitează realizarea tehnică a dispozitivelor destinate calculării funcțiilor logice cu un număr arbitrar de argumente.

Întrebări și exerciții

- 1 Determinați numărul funcțiilor logice de 5 și de 6 variabile.
- 2 Numiți funcțiile logice elementare și alcătuiți tabelele respective de adevăr.
- 3 Memorizați tabelele de adevăr ale funcțiilor logice frecvent utilizate *NU*, *ȘI*, *SAU*, *ȘI-NU*, *SAU-NU* și *COINCIDENTĂ*.
- 4 Elaborați un program care va afișa pe ecran tabelul de adevăr al uneia dintre funcțiile logice y_0, y_1, y_2, y_3 de 2 variabile.
- 5 Elaborați un program care va afișa pe ecran tabelul de adevăr al uneia dintre funcțiile logice y_j de n variabile.

5.1. Circuite logice elementare

Circuitul logic este un dispozitiv destinat calculării funcțiilor logice. Pentru a realiza circuitele logice, e necesar ca valorile binare **0** și **1** ale argumentelor și funcțiilor respective să fie reprezentate prin valorile unor mărimi fizice, de exemplu: presiune, temperatură, tensiune sau curent electric, flux luminos etc. În funcție de mărimile fizice utilizate, deosebim dispozitive logice mecanice, pneumatice, hidraulice, electromecanice, electronice, optice etc. În dispozitivele hidraulice și pneumatice valorile logice **0** sau **1** pot fi reprezentate prin valorile mici și, respectiv, mari ale presiunii fluidului, în dispozitivele electromecanice și electronice – prin prezența sau absența curentului electric, prin niveluri de tensiune etc.

Pentru o înțelegere clară a principiilor de funcționare a dispozitivelor logice, vom studia mai întâi circuitele cu contacte. Componentele de bază ale acestor circuite sunt **elementele de comutare** – contactele electrice normal deschise și contactele electrice normal închise.

În cazul **contactelor normal deschise**, circuitul electric este deschis dacă contactele nu sunt acționate și – închis la acționarea lor. În cazul **contactelor normal închise**, circuitul electric este închis dacă contactele nu sunt acționate și – deschis la acționarea lor (fig. 5.1).

	<i>Neacționate</i>	<i>Acționate</i>
Contacte normal deschise		
Contacte normal închise		

Fig. 5.1. Contacte normal deschise și normal închise

De exemplu, contactele unui întrerupător electric uzual sunt contacte normal deschise, iar contactele butonului de pauză al magnetofonului sunt contacte normal închise.

În circuitele cu contacte, valorile logice ale argumentelor sunt reprezentate prin stările contactelor electrice respective. Valorii logice **1** îi corespunde starea „contactul este acționat”, iar valorii logice **0** îi corespunde starea „contactul este neacționat”.

Circuitul electric care realizează **funcția logică NU** și simbolul utilizat sunt prezentate în *figura 5.2*.

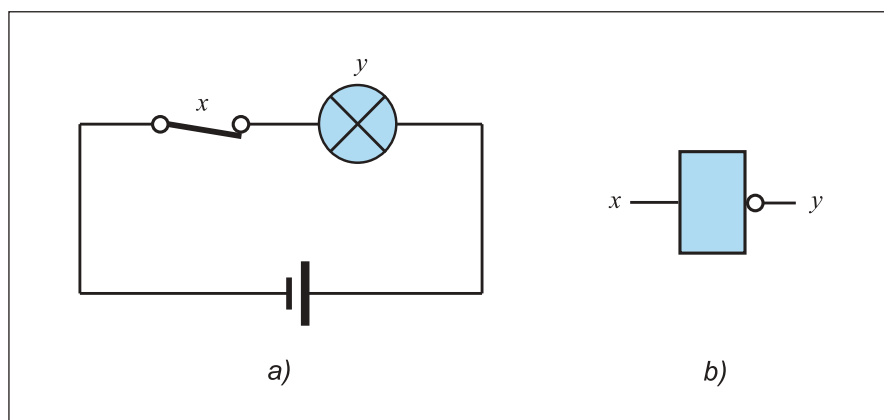


Fig. 5.2. Circuit cu contacte pentru realizarea funcției logice **NU** (a) și simbolul utilizat (b)

Argumentul x este materializat prin contactul normal închis, iar valorile variabilei dependente y sunt reprezentate prin stările becului electric: *stins* (valoarea logică **0**) sau *aprins* (valoarea logică **1**). Se observă că becul va fi aprins ($y=1$), dacă contactul normal închis nu este acționat ($x=0$).

Funcția logică ȘI se realizează prin conectarea în serie a contactelor electrice. Circuitul electric care realizează funcția **ȘI** de două variabile și simbolul utilizat sunt prezentate în *figura 5.3*.

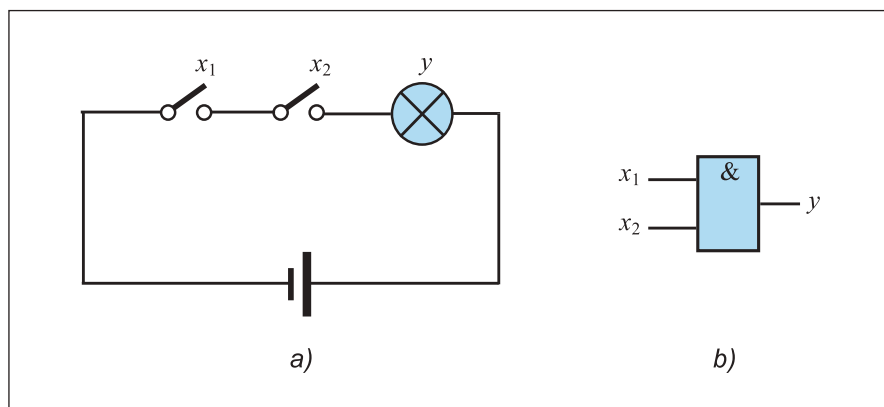


Fig. 5.3. Circuit cu contacte pentru realizarea funcției logice **ȘI** (a) și simbolul utilizat (b)

Variabilele x_1 și x_2 sunt materializate prin cele două contacte normale deschise, iar valorile variabilei y prin bec. Se observă că becul va fi aprins ($y=1$), numai dacă ambele contacte normale deschise sunt acționate ($x_1=1$ și $x_2=1$).

Funcția logică SAU se realizează prin conectarea în paralel a contactelor electrice. Circuitul respectiv este prezentat în *figura 5.4*, p. 144.

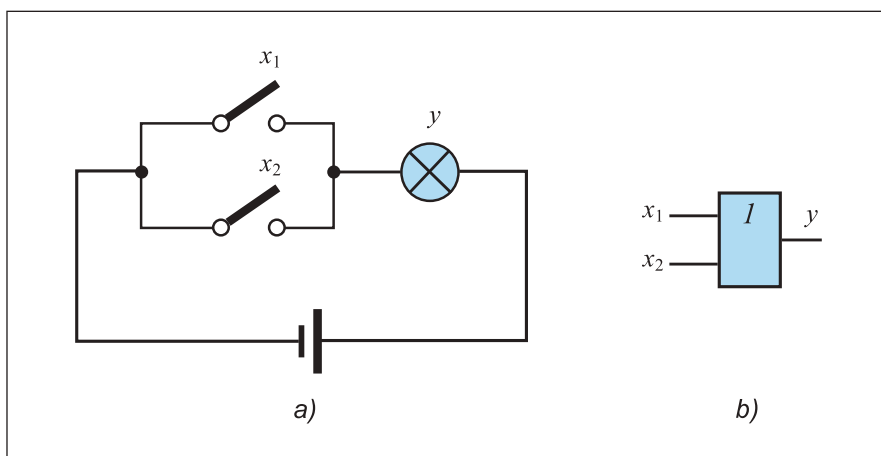


Fig. 5.4. Circuite cu contacte pentru realizarea funcției logice SAU (a) și simbolul utilizat (b)

Se constată că becul va fi aprins ($y=1$), dacă cel puțin unul dintre cele două contacte normale deschise este acționat ($x_1=1$ sau $x_2=1$).

Întrucât viteza de închidere-deschidere a contactelor electrice este foarte mică, în calculatoarele moderne valorile **0**, **1** sunt reprezentate prin niveluri de tensiune, iar ca element de comutare se utilizează tranzistorul.

Tranzistorul este un dispozitiv electronic format în sau pe suprafața unui monocristal semiconductor. Tehnologiile avansate permit fabricarea a 10^6 – 10^7 de tranzistoare pe o suprafață de 1 cm^2 al monocristalului.

În regim de comutație, tranzistorul poate fi considerat ca un întrerupător obișnuit, care într-o stare conduce curentul (este închis), iar în alta – nu (este deschis). Spre deosebire însă de întrerupătorul obișnuit, închiderea sau deschiderea tranzistorului se realizează cu ajutorul curentului electric.

Există mai multe tipuri de tranzistoare. În figura 5.5 este prezentat tranzistorul *n-p-n* (abrevierea se referă la structura internă a tranzistorului) și schemele echivalente în regim de comutație.

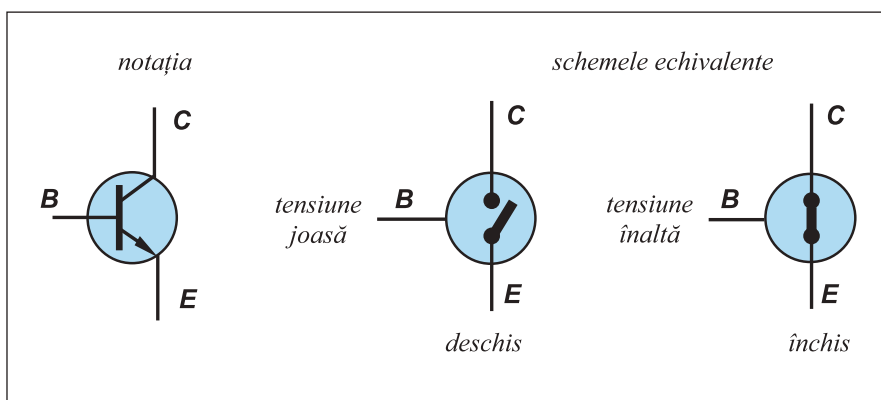


Fig. 5.5. Tranzistorul *n-p-n*

Tranzistorul $n-p-n$ are trei terminale: emitorul E , baza B și colectorul C . În regim de comutație, emitorul și colectorul pot fi considerați drept contacte care se închid sau se deschid cu ajutorul unei tensiuni aplicate la bază. Menționăm că tranzistoarele moderne permit efectuarea a 10^6 – 10^9 închideri-deschideri pe secundă. Ca și în cazul contactelor electrice studiate mai sus, utilizarea diferitor tipuri de tranzistoare și conectarea lor în serie sau în paralel permite realizarea funcțiilor logice NU , ȘI , SAU .

Circuitele destinate calculării funcțiilor logice frecvent utilizate se numesc circuite logice elementare sau porți logice.

Simbolurile utilizate pentru notarea porților logice sunt reprezentate în figura 5.6.

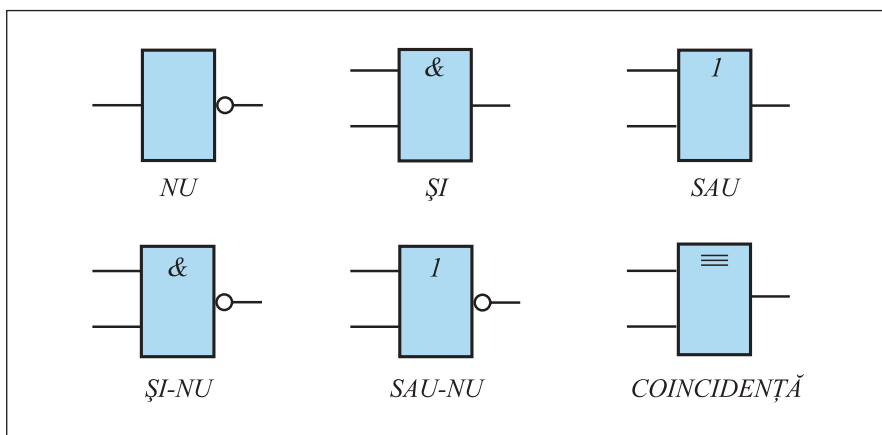


Fig. 5.6. Simbolurile porților logice

E cunoscut faptul că orice funcție logică poate fi exprimată printr-o formulă care include numai simbolurile operatorilor elementari $\bar{}$, $\&$, $\vee$. Prin urmare, **orice funcție logică cu un număr arbitrar de argumente poate fi materializată prin conectarea porților logice NU , ȘI , SAU .** De exemplu, funcția

$$y = x_1 x_2 \vee \bar{x}_2 x_3$$

poate fi realizată cu ajutorul următoarelor porți logice:

- o poartă NU pentru calcularea $\bar{x}_2$;
- două porți logice ȘI pentru calcularea conjuncțiilor $x_1 x_2$ și $\bar{x}_2 x_3$;
- o poartă SAU pentru calcularea disjuncției $x_1 x_2 \vee \bar{x}_2 x_3$;

Schema circuitului logic pentru calcularea funcției respective este prezentată în figura 5.7, p. 146.

Întrebări și exerciții

- ❶ Cum pot fi reprezentate valorile binare **0** și **1**?
- ❷ Cum funcționează contactele normal deschise și contactele normal închise?
- ❸ Care este reprezentarea valorilor binare **0** și **1** în circuitele cu contacte?
- ❹ EXPERIMENTEAZĂ! Utilizând trusa din laboratorul de fizică, montați cir-

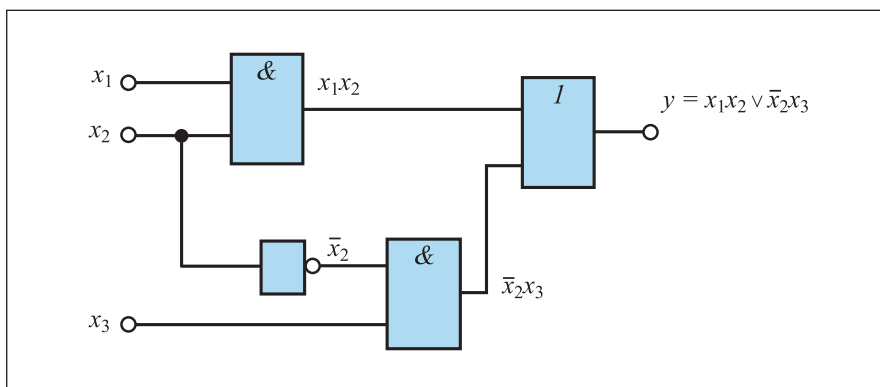


Fig. 5.7. Circuitul logic pentru realizarea funcției $y = x_1x_2 \vee \bar{x}_2x_3$

cuitele din figurile 5.2, 5.3 și 5.4. Verificați tabelele de adevăr ale funcțiilor realizate de circuitele în cauză.

- 5 Cum se realizează funcțiile *NU*, *ȘI*, *SAU* în cazul circuitelor cu contacte electrice?
- 6 Care este rolul elementelor de comutare la realizarea circuitelor logice?
- 7 Care este rolul tranzistorului în calculatoarele moderne?
- 8 Memorizați simbolurile porților logice. Explicați cum se utilizează porțile logice la realizarea funcțiilor logice arbitrare.
- 9 PROIECTEAZĂ! Utilizând porțile *NU*, *ȘI*, *SAU*, elaborați circuitele logice pentru calcularea următoarelor funcții:

- | | |
|---|--|
| a) $y = x_1x_2x_3;$ | i) $y = x_1x_2 \vee \bar{x}_2\bar{x}_3;$ |
| b) $y = x_1 \vee x_2 \vee x_3;$ | j) $y = (x_1 \vee x_2) (\bar{x}_2 \vee \bar{x}_3);$ |
| c) $y = \bar{x}_1x_2x_3;$ | k) $y = x_1x_2 \vee \bar{x}_1x_3 \vee x_3x_4;$ |
| d) $y = \bar{x}_1 \vee x_2 \vee x_3;$ | l) $y = \bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3;$ |
| e) $y = x_1x_2 \vee x_3x_4;$ | m) $y = \bar{x}_1x_2 \vee x_1\bar{x}_2;$ |
| f) $y = (x_1 \vee x_2) (x_3 \vee x_4);$ | n) $y = x_1x_2 \vee \bar{x}_1\bar{x}_2;$ |
| g) $y = x_1x_2;$ | o) $y = x_1(x_2 \vee x_3 \vee x_4);$ |
| h) $y = x_1 \vee x_2;$ | p) $y = x_1 \vee x_2x_3x_4.$ |

- 10 CERCETEază! Releul electromagnetic este un dispozitiv cu care se comandă închiderea sau deschiderea contactelor electrice. Contactele respective sunt acționate de un electromagnet. Cum poate fi utilizat releul pentru realizarea funcțiilor logice *NU*, *ȘI*, *SAU*?

EXPERIMENTEAZĂ! Cu piesele din trusa din laboratorul de fizică asamblați un releu electromagnetic și verificați soluțiile propuse de dvs.

- 11 EXPLOREAZĂ! Reprezentând valorile binare ale variabilelor de ieșire prin prezența (valoarea 1) sau absența (valoarea 0) a fluidului, elaborați circuitele hidraulice pentru realizarea funcțiilor logice *NU*, *ȘI*, *SAU*. Montați instalațiile respective utilizând robinetele și furtunurile din trusa din laboratorul de chimie. Verificați tabelele de adevăr ale funcțiilor logice realizate.

5.2. Clasificarea circuitelor logice

Circuitele logice se clasifică în două categorii: circuite combinaționale și circuite secvențiale.

Într-un **circuit combinațional** valorile variabilelor de ieșire sunt determinate în orice moment de combinația valorilor variabilelor de intrare conform funcțiilor logice ale circuitului. Într-un **circuit secvențial** valorile variabilelor de ieșire depind nu numai de combinațiile valorilor variabilelor de intrare, dar și de consecutivitatea aplicării lor. Altfel spus, circuitele combinaționale reprezintă circuite logice lipsite de elemente de memorie, iar circuitele secvențiale includ și elementele de memorie binară. Prin urmare, un circuit combinațional realizează o prelucrare numerică a informației, care poate fi în întregime exprimată printr-un grup de funcții logice în care nu intervine parametrul timp.

Schema-bloc a unui circuit combinațional este prezentată în figura 5.8.

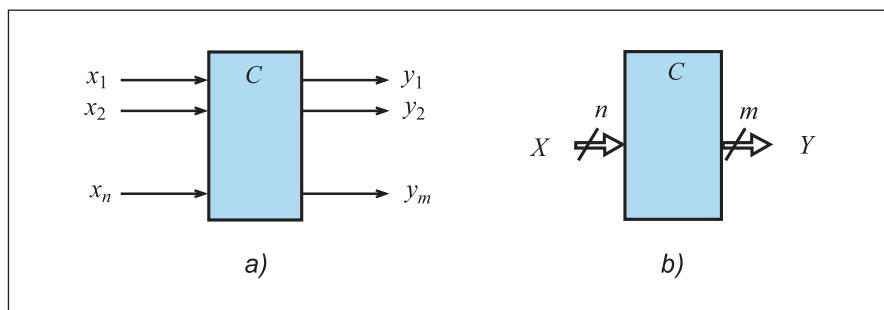


Fig. 5.8. Schema-bloc a unui circuit logic combinațional:
a – detaliat; b – generalizat

Circuitul are n variabile de intrare: $X = \langle x_1, x_2, \dots, x_n \rangle$, și m variabile de ieșire: $Y = \langle y_1, y_2, \dots, y_m \rangle$. Conexiunile destinate transmiterii valorilor unui grup de variabile se reprezintă prin linii duble. Dacă e necesar, numărul de variabile se indică lângă o bară care întretaie linia dublă a grupului de variabile. De exemplu, în figura 5.8 b se indică că grupul X include n variabile, iar grupul Y – m variabile.

Din punctul de vedere al teoriei informației, circuitul combinațional reprezintă un convertor de cod: la intrările X se aplică combinațiile admise de primul cod, din care se face conversiunea, iar la ieșirile Y apar combinațiile corespunzătoare în cel de al doilea cod, în care se face conversiunea.

De exemplu, primul cod poate fi codul *EBCDIC*, iar al doilea codul *ASCII*.

5.3. Sumatorul

Una dintre principalele sarcini ale unui calculator în momentul prelucrării informației constă în efectuarea operațiilor aritmetice elementare și, în special, adunarea și scăderea. Dispozitivele în care au loc aceste operații au la bază circuite cu ajutorul cărora se efectuează adunarea, respectiv scăderea a două cifre binare.

Semisumatorul este un circuit combinațional destinat adunării a două cifre binare. Tabelul de adevăr care pune în evidență funcționarea unui semisumator rezultă din regula de adunare a două cifre binare și este prezentat în *figura 5.9*. Aici a și b reprezintă cele două cifre binare care se adună, s – cifra-sumă a rangului respectiv, iar t – cifra de transport către rangul următor.

a	b	s	t
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig. 5.9. Tabelul de adevăr pentru adunarea a două cifre binare

Pentru a elabora o schemă posibilă a semisumatorului, exprimăm funcțiile de ieșire s și t :

$$s = \bar{a}b \vee a\bar{b};$$

$$t = ab.$$

Schema care realizează funcțiile s , t și simbolul utilizat sunt prezentate în *figura 5.10*.

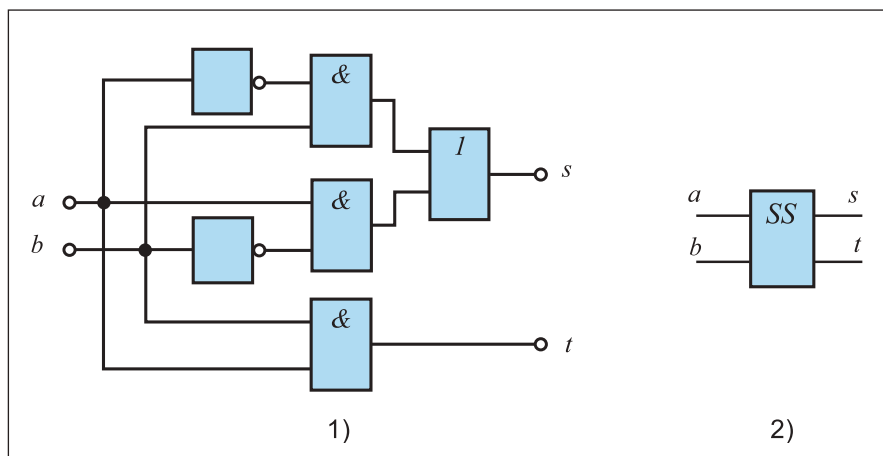


Fig. 5.10. Schema semisumatorului (1) și simbolul utilizat (2)

Să luăm două numere binare

$$A = a_{n-1} a_{n-2} \dots a_j \dots a_0$$

și

$$B = b_{n-1} b_{n-2} \dots b_j \dots b_0,$$

unde a_j și b_j reprezintă cifrele binare din rangul (poziția) j . La însumarea cifrelor a_j și b_j din rangul j trebuie să se ia în considerare și cifra de transport t_{j-1} de la rangul inferior $j-1$:

$$\begin{array}{r}
 t_{j-1} \\
 a_{n-1} a_{n-2} \dots a_j \dots a_0 \\
 + \\
 b_{n-1} b_{n-2} \dots b_j \dots b_0
 \end{array}$$

Se obține astfel un circuit combinațional care calculează suma $t_{j-1} + a_j + b_j$, denumit **sumator elementar**.

Sumatorul elementar poate fi realizat prin conectarea în cascadă a două semisumatoare SS_1 și SS_2 (fig. 5.11).

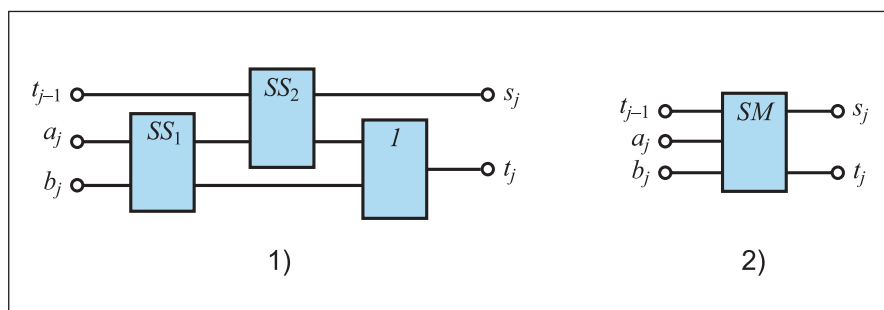


Fig. 5.11. Schema sumatorului elementar (1) și simbolul utilizat (2)

Semisumatorul SS_1 calculează suma $(a_j + b_j)$, iar semisumatorul SS_2 însumează transportul t_{j-1} cu suma $(a_j + b_j)$ calculată de primul semisumator. Transportul t_j către rangul superior $j + 1$ se calculează de poarta logică SAU , care reunește transporturile intermediare de la ieșirile respective ale semisumatoarelor SS_1 și SS_2 .

Suma numerelor binare A și B se calculează cu ajutorul unui circuit combinațional denumit **sumator**. Un sumator poate fi realizat prin conectarea a n sumatoare elementare (fig. 5.12, p. 150).

Sumatorul elementar SM_0 corespunzător cifrei celei mai puțin semnificative poate fi înlocuit cu un semisumator, deoarece pentru această poziție nu există un transport de la rangul precedent. Transportul de la ieșirea sumatorului SM_{n-1} al rangului cel mai semnificativ este folosit pentru a indica **depășirea capacității** sumatorului de n biți.

Din analiza figurilor 5.10, 5.11 și 5.12 rezultă că un dispozitiv complex – sumatorul de n biți, este realizat prin reunirea a unor dispozitive mult mai simple, adică a n sumatoare elementare. Fiecare sumator elementar, la rândul lui, este realizat prin reunirea a câte două semisumatoare și o poartă logică SAU .

Metoda de elaborare a dispozitivelor complexe (de exemplu, sumatorul) prin reunirea mai multor dispozitive identice mai simple (sumatorul elementar) poartă denumirea de **metodă de proiectare ierarhică**. Conform acestei metode, componentele calculatorului se caracterizează prin niveluri de ierarhie, de exemplu:

- nivelul 1 – tranzistoare;
- nivelul 2 – porți logice;
- nivelul 3 – semisumatoare, sumatoare elementare etc.;
- nivelul 4 – sumatoare, scăzătoare etc.;
- nivelul 5 – unități aritmetice, unități de comandă etc.

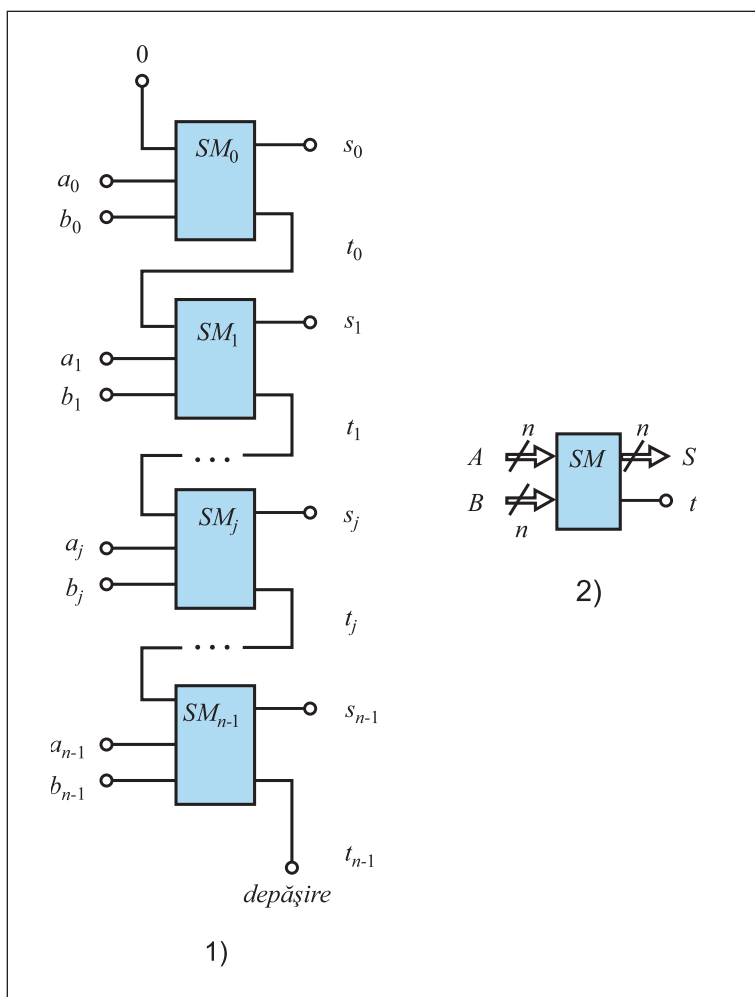


Fig. 5.12. Schema sumatorului (1) și simbolul utilizat (2)

Componentele de un nivel ierarhic inferior sunt utilizate în calitate de „cubșoa-re” elementare pentru realizarea componentelor de un nivel ierarhic superior.

Teoretic, dispozitivele unui calculator numeric pot fi elaborate și fără aplicarea metodei de proiectare ierarhică. De exemplu, în cazul sumatorului elementar, utilizarea semisumatorului nu este obligatorie. E suficient să alcătuim tabelul de adevăr al sumatorului elementar, să exprimăm funcțiile de ieșire prin formule și să reunim porțile logice respective.

În cazul nivelurilor ierarhice superioare, neaplicarea metodei de proiectare ierarhică face imposibilă elaborarea dispozitivelor complexe. De exemplu, în cazul unui sumator de n biți, tabelul respectiv de adevăr ar conține 2^{2n} rânduri. Pentru $n = 16$ obținem $2^{2 \cdot 16} = 2^{32} \approx 10^9$ rânduri. Evident, formulele funcțiilor de ieșire ale sumatorului de 16 biți practic nu mai pot fi scrise. Prin urmare, suntem nevoiți să aplicăm metoda de proiectare ierarhică și să realizăm sumatorul printr-o reunire de n sumatoare elementare.

Aplicând metoda de proiectare ierarhică, într-un mod similar pot fi elaborate circuitele combinaționale destinate scăderii numerelor binare: **semiscăzătorul**, **scăzătorul elementar** și **scăzătorul**.

Întrebări și exerciții

- ❶ Care este destinația semisumatorului? Dar a sumatorului elementar? Și în fine a sumatorului de n biți?
- ❷ CREEAZĂ! Alcătuiți tabelul de adevăr al sumatorului elementar. Tabelul va conține cinci coloane: trei pentru intrările a_j , b_j , t_{j-1} și două pentru ieșirile s_j , t_j .
- ❸ Elaborați un program care simulează funcționarea sumatorului elementar. Cifrele binare a_j , b_j și cifra de transport t_{j-1} de la rangul inferior se citesc de la tastatură, iar cifra-sumă s_j și cifra de transport t_j către rangul superior se afișează pe ecran.
- ❹ Explicați esența metodei de proiectare ierarhică a dispozitivelor unui calculator numeric. Este oare obligatorie aplicarea acestei metode? Argumentați răspunsul dvs.
- ❺ Câte porți logice *NU*, *ȘI*, *SAU* va conține un sumator de 16 biți? Dar de 32 de biți?
- ❻ PROIECTEAZĂ! **Semiscăzătorul** este un circuit combinațional destinat scăderii a două cifre binare. Circuitul respectiv are intrările a , b și ieșirile d , i . Prin termenul d se înțelege diferența $a-b$, iar prin termenul i – împrumutul de la cifra de rang imediat superior. Alcătuiți tabelul de adevăr și elaborați schema semiscăzătorului.
- ❼ PROIECTEAZĂ! **Scăzătorul elementar** este un circuit combinațional capabil de a calcula diferența d_j și împrumutul i_j către rangul imediat superior, dacă se introduce la intrare descăzutul a_j , scăzătorul b_j și împrumutul i_{j-1} de la rangul anterior. Aplicând metoda de proiectare ierarhică, elaborați schema scăzătorului elementar.
- ❽ PROIECTEAZĂ! Aplicând metoda de proiectare ierarhică, elaborați schema unui **scăzător** de n biți.
- ❾ Câte porți logice *NU*, *ȘI*, *SAU* va conține un scăzător de 16 biți? Dar de 32 de biți?
- ❿ ANALIZEAZĂ! Este oare obligatorie aplicarea metodei de proiectare ierarhică în cazul elaborării unui scăzător de n biți? Argumentați răspunsul dvs.

5.4. Circuite combinaționale frecvent utilizate

Circuitele combinaționale frecvent utilizate sunt prezentate în *figura 5.13*, p. 152.

Sumatorul este un circuit combinațional destinat adunării a două numere binare. Tabelul de adevăr și schema sumatorului au fost studiate în paragraful precedent.

Comparatorul este un circuit combinațional care compară numerele binare A și B , indicând prin cele trei ieșiri una dintre situațiile posibile: $A < B$, $A > B$ sau $A = B$.

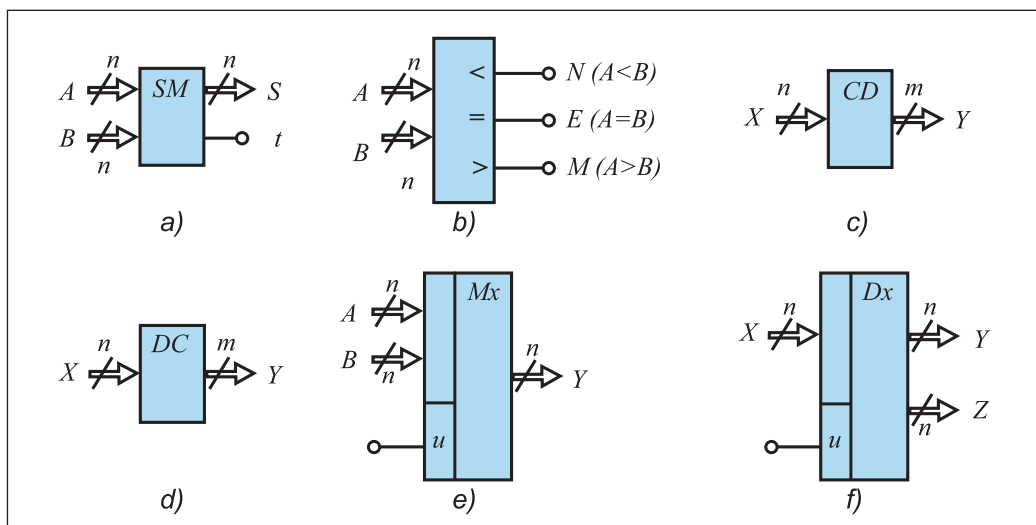


Fig. 5.13. Circuite combinaționale frecvent utilizate:
a – sumatorul; b – comparatorul; c – codificatorul; d – decodificatorul;
e – multiplexorul; f – demultiplexorul

Codificatorul este un circuit combinațional care efectuează conversiunea mesajelor $s_1, s_2, \dots, s_n$ în cuvintele binare din codul respectiv. Se consideră că fiecare dintre mesajele s_i este reprezentat prin valorile $x_1 = 0, \dots, x_i = 1, \dots, x_n = 0$ aplicate la intrarea codificatorului, iar cuvântul de cod – prin variabilele de ieșire $y_1, y_2, \dots, y_m$.

De exemplu, variabilele $x_1, x_2, x_3, \dots$ pot reprezenta starea tastelor <A>, , <C>, ... ale tastaturii. Codificatorul respectiv va furniza la ieșire cuvântul de cod ASCII corespunzător tastei acționate.

Decodificatorul este un circuit combinațional care generează semnalul logic 1 pe o ieșire distinctă pentru fiecare combinație a valorilor variabilelor de intrare. Cu alte cuvinte, decodificatorul efectuează operația inversă a unui codificator.

Decodificatoarele sunt utilizate pentru a determina operațiile pe care trebuie să le execute procesorul, pentru selectarea unităților de intrare-ieșire, sintetizarea simbolurilor etc.

Multiplexorul este un circuit combinațional destinat selectării fluxurilor de date. În figura 5.13e este prezentat un multiplexor care transmite la ieșire biții numărului binar A ($u = 0$) sau B ($u = 1$). În calculatoarele moderne multiplexoarele se utilizează pentru transferul informației de la mai multe surse la un singur destinatar.

Demultiplexorul distribuie fluxul de date de la intrarea X la una dintre ieșirile Y ($u = 0$) sau Z ($u = 1$). Drept exemplu amintim transferul informației de la o singură sursă la mai mulți destinatari.

Întrebări și exerciții

- 1 Explicați destinația circuitelor combinaționale frecvent utilizate: sumatorul, comparatorul, codificatorul, decodificatorul, multiplexorul și demultiplexorul.
- 2 Alcătuiți tabelul de adevăr al unui comparator de 2 biți.

③ Câte intrări și câte ieșiri poate avea un codificator? Câte intrări și câte ieșiri poate avea un decodificator?

④ CREEAZĂ! Pe panoul de comandă al imprimantei sunt montate butoanele *ON LINE* (funcționare sub controlul unității centrale), *OFF LINE* (funcționare autonomă), *LINE FEED* (avans de linie) și *FORM FEED* (avans de pagină). Alcătuiți tabelul de adevăr al codicatorului care furnizează la ieșire următoarele combinații binare:

00 – *ON LINE*;

01 – *OFF LINE*;

10 – *LINE FEED*;

11 – *FORM FEED*.

⑤ CREEAZĂ! Pe panoul de comandă al imprimantei sunt montate becurile (diodele luminescente) indicatoare *READY* (disponibilă), *PAPER* (lipsa de hârtie), *TEST* (regimul de testare) și *LOAD* (regimul de încărcare a informației). Alcătuiți tabelul de adevăr al decodicatorului care aprinde becurile în cauză. Stările respective sunt codificate prin următoarele combinații binare:

00 – *READY*;

01 – *PAPER*;

10 – *TEST*;

11 – *LOAD*.

⑥ CREEAZĂ! Tastatura calculatorului include tastele funcționale <*F1*>, <*F2*>, ..., <*F12*>. Alcătuiți tabelul de adevăr al codicatorului care va furniza la ieșire numărul binar corespunzător tastei funcționale acționate.

⑦ CREEAZĂ! Unitățile de intrare-ieșire ale unui calculator au următoarele adrese:

0000 – tastatura;

0001 – vizualizatorul;

0010 – imprimanta mecanică;

0011 – imprimanta cu jet de cerneală;

0100 – imprimanta laser;

0101 – unitatea de disc flexibil *A*;

0110 – unitatea de disc flexibil *B*;

0111 – unitatea de disc rigid *C*;

1000 – unitatea de disc rigid *D*.

Alcătuiți tabelul de adevăr al decodicatorului care va selecta unitatea indicată de adresa respectivă.

⑧ CREEAZĂ! Operațiile aritmetice ale unui calculator ipotetic sunt codificate după cum urmează:

<i>adunarea</i>	– 000;
<i>scăderea</i>	– 001;
<i>înmulțirea</i>	– 010;
<i>împărțirea</i>	– 011;
<i>compararea</i>	– 100.

Alcătuți tabelul de adevăr al decodificatorului operațiilor aritmetice.

- 9 CREEAZĂ! Alcătuți tabelul de adevăr al multiplexorului cu 2 linii de intrare.

5.5. Bistabilul RS

E cunoscut faptul că într-un circuit secvențial valorile variabilelor de ieșire depind nu numai de combinațiile valorilor variabilelor de intrare, dar și de **consecutivitatea aplicării lor**. Cu alte cuvinte, circuitul secvențial memorizează informații referitoare la combinațiile aplicate la intrări în momentele precedente. Acest lucru este posibil datorită faptului că circuitele secvențiale sunt alcătuite din circuite combinaționale și elemente de **memorie binară**.

Elementul de memorie binară este un circuit cu două stări distincte, destinat pentru a păstra o informație dintr-un singur bit. Circuitul respectiv are denumirea de bistabil.

În figura 5.14 este prezentată schema circuitului bistabil de bază realizat cu porți logice SAU-NU, denumit **bistabil asincron RS**.

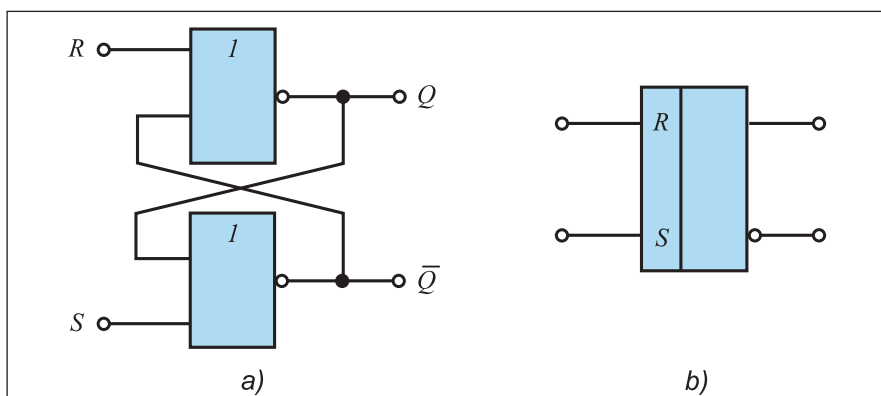


Fig. 5.14. Schema bistabilului asincron RS (a) și simbolul utilizat (b)

Circuitul are două intrări notate cu R și S și două ieșiri notate cu Q și $\bar{Q}$. Se observă că semnalele de ieșire Q și $\bar{Q}$ sunt aplicate la intrările porților SAU-NU. Conexiunile respective sunt numite **reacții**. Tocmai datorită acestor conexiuni, circuitul dat are două stări distincte și, prin urmare, asigură memorarea unui bit de informație.

Într-adevăr, admitem că semnalele de intrare $R = S = 0$, iar ieșirile $Q = 1, \overline{Q} = 0$. Datorită reacției, valoarea $Q = 1$ impune ieșirea $\overline{Q} = 0$. La rândul ei, tot datorită reacției, valoarea $\overline{Q} = 0$ confirmă ieșirea $Q = 1$. Exact la fel, în cazul în care ieșirile $Q = 0, \overline{Q} = 1$, datorită reacției, valoarea $\overline{Q} = 1$ impune ieșirea $Q = 0$. Prin urmare, valorile de intrare $R = S = 0$ nu schimbă starea bistabilului, acesta păstrând cifra binară memorată.

S-a convenit ca stării $Q = 1, \overline{Q} = 0$ să-i corespundă cifra binară 1, iar stării $Q = 0, \overline{Q} = 1$ cifra binară 0. Ieșirea Q este numită ieșire directă sau ieșire adevărată, iar ieșirea $\overline{Q}$ – ieșire inversă sau ieșire negată. Starea bistabilului este indicată de ieșirea directă Q .

Să examinăm cazul în care $R = 0, S = 1$. Presupunem că bistabilul se află în starea 0, adică $Q = 0$ și $\overline{Q} = 1$. În acest caz, valoarea $S = 1$ va impune $\overline{Q} = 0$, care, la rândul ei, prin reacție, va asigura $Q = 1$. Prin urmare, bistabilul trece în starea 1 ($Q = 1, \overline{Q} = 0$). Această stare, după cum s-a stabilit anterior, se va păstra și după trecerea semnalului S de la valoarea 1 la valoarea 0. Intrarea S , care asigură fixarea bistabilului în starea 1, poartă denumirea de **intrare de setare**.

În mod similar se poate constata că în cazul în care bistabilul se află în starea 1 ($Q = 1, \overline{Q} = 0$), iar $S = 0$, și $R = 1$, bistabilul va trece în starea 0 ($Q = 0, \overline{Q} = 1$). Această stare se va păstra și după trecerea semnalului R de la valoarea 1 la valoarea 0. Intrarea R , care asigură stabilirea bistabilului în starea 0, poartă denumirea de **intrare de resetare**.

Combinăția de intrare $R = 1$ și $S = 1$ impune valorile de ieșire $Q = 0$ și $\overline{Q} = 0$. Valorilor de ieșire $Q = \overline{Q} = 0$ nu le corespunde nicio stare distinctă a bistabilului. Prin urmare, pentru bistabilul RS combinația de intrare $R = S = 1$ este o **combinație interzisă**.

Regimurile de funcționare a bistabilului în cauză sunt totalizate în *tabelul 5.1*.

Tabelul 5.1

Regimurile de funcționare a bistabilului asincron RS

Intrări $R \quad S$		Regim de funcționare	Ieșirea Q
0	0	păstrare bit	bit memorat
0	1	setare	1
1	0	resetare	0
1	1	interzis	–

Adjectivul *asincron* din denumirea bistabilului RS concretizează modul de influență a semnalelor de comandă R și S asupra stării bistabilului. Din analiza circuitului prezentat în *figura 5.14* rezultă că semnalele de comandă, aplicate la intrările R și S în momente arbitrare, pot schimba starea bistabilului.

Circuitele secvențiale în care starea circuitului poate fi schimbată de semnalele de comandă în momente arbitrare de timp se numesc circuite asincrone.

Un calculator modern conține zeci de mii de bistabile. Schimbarea stării lor în momente arbitrare, greu de controlat, poate fi cauza unor erori de funcționare. Pentru a evita o funcționare greșită, s-a convenit ca această comportare a circuitelor secvențiale să fie determinată în baza valorilor semnalelor de comandă aplicate la intrări în momente discrete, bine determinate în timp. Aceste momente sunt indicate cu ajutorul unor impulsuri speciale, numite **semnale de sincronizare**.

Circuitele secvențiale în care starea circuitului poate fi schimbată de semnalele de comandă numai în momentele indicate de semnalele de sincronizare se numesc circuite sincrone.

De obicei, semnalul de sincronizare se notează prin litera C (din engleză *Clock* „ceas”) și este furnizat de un dispozitiv special, numit **ceas de sistem**.

În figura 5.15 este prezentată schema unui bistabil sincron denumit **bistabil sincron RS**.

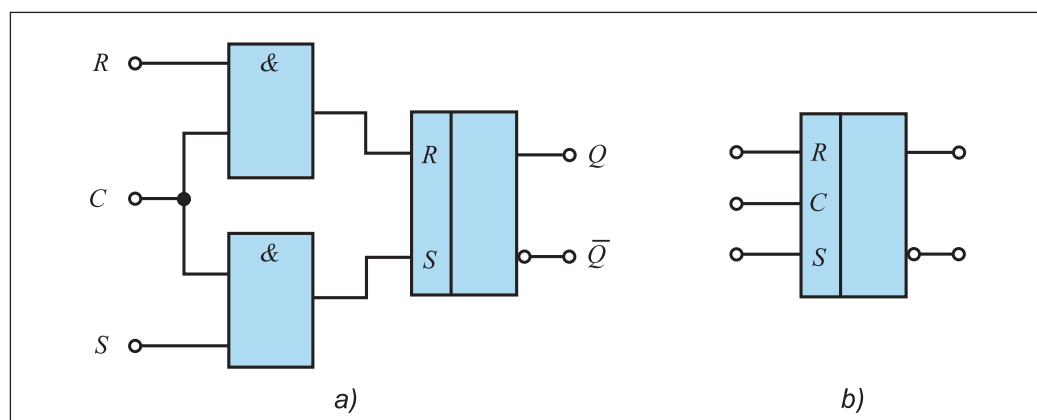


Fig. 5.15. Schema bistabilului sincron RS (a) și simbolul utilizat (b)

Schema dată include bistabilul asincron RS (fig. 5.14) și două porți logice ȘI care permit aplicarea semnalelor de comandă la intrările bistabilului asincron numai în cazul în care semnalul de sincronizare C are valoarea 1.

Întrebări și exerciții

- ❶ Prin ce se deosebesc circuitele combinaționale și circuitele secvențiale?
- ❷ Care este destinația circuitului bistabil?
- ❸ Cum funcționează un circuit bistabil cu porți logice SAU-NU? Care este destinația reacțiilor?
- ❹ Explicați regimurile de funcționare a bistabilului asincron RS. De ce combinația $R = S = 1$ nu poate fi aplicată la intrările bistabilului examinat?
- ❺ STUDIU DE CAZ. Prin ce se deosebesc circuitele secvențiale asincrone și circuitele secvențiale sincrone?
- ❻ Explicați cum funcționează bistabilul sincron RS. Care este destinația porților logice ȘI din componența acestui bistabil?

- 7 În figura 5.16 este prezentată schema circuitului bistabil de bază realizat cu porți logice ȘI-NU, denumit **bistabil asincron $\overline{R}\overline{S}$** .

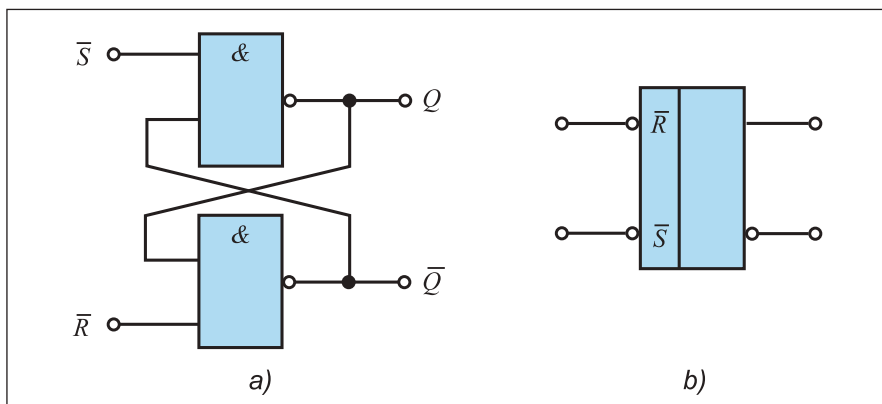


Fig. 5.16. Schema bistabilului asincron $\overline{R}\overline{S}$ (a) și simbolul utilizat (b)

Circuitul are următoarele regimuri de funcționare:

- păstrare bit ($\overline{R} = 1, \overline{S} = 1$);
- setare ($\overline{R} = 1, \overline{S} = 0$);
- resetare ($\overline{R} = 0, \overline{S} = 1$).

Explicați cum funcționează bistabilul examinat. De ce combinația de intrare $\overline{R} = 0, \overline{S} = 0$ este o combinație interzisă?

- 8 ELABOREAZĂ! Utilizând bistabilul asincron $\overline{R}\overline{S}$, elaborați schema **bistabilului sincron $\overline{R}\overline{S}$** .
- 9 PROIECTEAZĂ! Desenați schema detaliată (la nivelul porților logice ȘI, SAU-NU) a bistabilului sincron $\overline{R}\overline{S}$. Pentru aceasta decalcați schemele respective din figurile 5.14a și 5.15a.
- 10 PROIECTEAZĂ! Desenați schema detaliată (la nivelul porților logice ȘI, ȘI-NU) a bistabilului sincron $\overline{R}\overline{S}$.

5.6. Circuite secvențiale frecvent utilizate

Registrul (fig. 5.17a, p. 158) este un dispozitiv numeric destinat păstrării temporare a unui număr binar,

$$D = d_{n-1} \dots d_1 d_0.$$

Registrul este constituit din bistabile la care sunt atașate circuite combinaționale ce permit înscrierea, citirea sau transferul informației. Înscrierea informației în registru se efectuează prin aplicarea la intrarea W (Write „înscrie”) a impulsului respectiv. Deoarece fiecare bistabil poate memora un singur bit, **capacitatea n** a unui registru este dată de numărul bistabilelor.

În anumite aplicații, de exemplu: înmulțirea și împărțirea numerelor binare, scrierea sau citirea datelor pe disc, transmiterea datelor prin firele telefonice etc., apare necesitatea deplasării în stânga sau în dreapta a informației memorate într-un

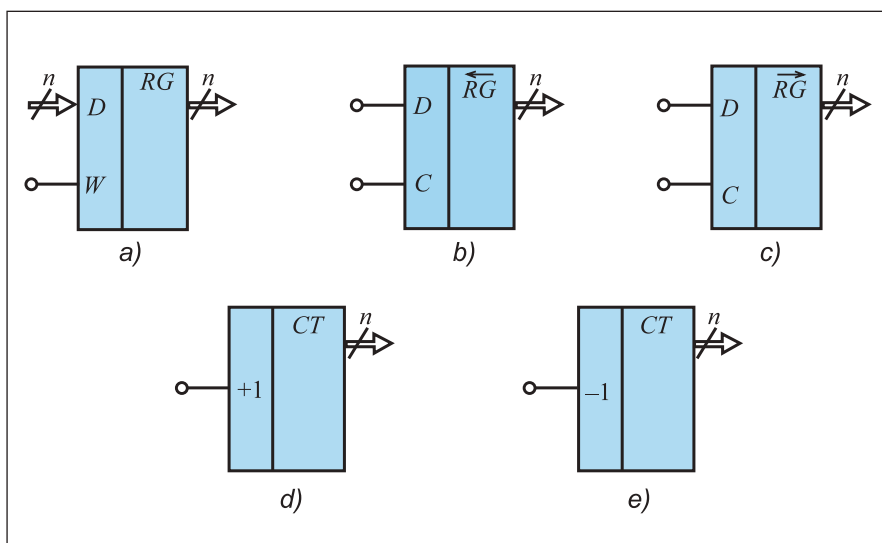


Fig. 5.17. Circuite secvențiale frecvent utilizate:

a – registru; b – registru de deplasare de la dreapta spre stânga; c – registru de deplasare de la stânga spre dreapta; d – numărător direct; e – numărător invers

registru. În acest scop se utilizează **registrele de deplasare** (fig. 5.17 b, c).

Sucesiunea stărilor unui **registru de deplasare de la dreapta spre stânga** este dată în figura 5.18.

<i>Timpul</i>	d_3	d_2	d_1	d_0
inițial	0	1	0	1
t_1	1	0	1	0
t_2	0	1	0	0
t_3	1	0	0	0
t_4	0	0	0	0
t_5	0	0	0	0
...			...	

Fig. 5.18. Sucesiunea stărilor unui registru de deplasare de la dreapta spre stânga

Se consideră că starea inițială a registrului este 0101, iar impulsurile sunt aplicate la intrarea C în momentele de timp t_1, t_2, t_3 etc. Firește, registrul în cauză calculează produsul $D \times 2$.

În mod similar, un **registru de deplasare de la stânga spre dreapta** va calcula câtul $D : 2$.

Numărătoarele sunt circuite secvențiale care înregistrează numărul de impulsuri aplicat la intrare. Numărătoarele se clasifică după următoarele criterii:

- modul de codificare a informației (binar, binar-zecimal etc.);
- modul de modificare a stărilor numărătorului (numărătoare directe, inverse și reversibile).

În general, numărătoarele sunt realizate prin asocierea circuitelor bistabile cu circuite combinaționale care determină modul corect în care numărătoarele urmează să-și modifice starea la fiecare nou impuls sosit la intrare.

Un **numărător binar** înregistrează succesiunea impulsurilor aplicate la intrare în sistemul de numerație binar. **Capacitatea de numărare** a numărătorului binar depinde de numărul bistabilelor. Considerând pentru fiecare număr o stare distinctă, rezultă că acesta poate număra în gama 0 la 2^n-1 , n fiind numărul bistabilelor.

În figura 5.17d este prezentat **numărătorul binar direct**, iar în figura 5.19 tabelul succesiunii stărilor unui numărător binar direct de 3 biți.

Numărătoarele care își schimbă starea conform tabelului din figura 5.19 sunt

<i>Timpul</i>	<i>d₂</i>	<i>d₁</i>	<i>d₀</i>
inițial	0	0	0
<i>t</i> ₁	0	0	1
<i>t</i> ₂	0	1	0
<i>t</i> ₃	0	1	1
<i>t</i> ₄	1	0	0
<i>t</i> ₅	1	0	1
<i>t</i> ₆	1	1	0
<i>t</i> ₇	1	1	1
<i>t</i> ₈	0	0	0
<i>t</i> ₉	0	0	1
...	...		

Fig. 5.19. Succesiunea stărilor unui numărător binar direct de trei biți

denumite **directe**, deoarece conținutul numărătorului crește cu o unitate, la fiecare nou impuls sosit la intrarea +1. Dacă într-un numărător se introduce inițial un număr și la fiecare impuls aplicat la intrarea -1 conținutul său scade cu o unitate, se obține un **numărător invers** (fig. 5.17e).

Întrebări și exerciții

- ❶ Care este destinația registrului? De ce depinde capacitatea unui registru?
- ❷ ANALIZEAZĂ! În registrul de deplasare de la dreapta spre stânga (fig. 5.17b) este încărcat numărul binar 1001. Care va fi conținutul registrului după aplicarea la intrarea C a unui impuls? Dar a două impulsuri?
- ❸ În registrul de deplasare de la stânga spre dreapta este încărcat unul dintre următoarele cuvinte binare:

a) 00000;

b) 10000;

c) 01000;

d) 00100;

e) 00010;

f) 00001;

- g) 10001; i) 01100;
 h) 01010; j) 00110.

Care va fi conținutul registrului după aplicarea la intrarea C a două impulsuri consecutive?

- ④ **PROGRAMEAZĂ!** Elaborați un program care simulează funcționarea unui registru de deplasare de la stânga spre dreapta de n biți.
 ⑤ **STUDIU DE CAZ.** Care este destinația numărătoarelor? Cum se schimbă stările unui numărător direct? Dar ale unui numărător invers?
 ⑥ **ANALIZEAZĂ!** Un numărător direct de 4 biți se află inițial în una dintre următoarele stări:

- a) 0000; f) 1010;
 b) 0010; g) 1100;
 c) 0100; h) 1111;
 d) 1000; i) 0101;
 e) 1001; j) 0110.

Care va fi starea numărătorului după aplicarea a 5 impulsuri de intrare? Dar a 8 impulsuri?

- ⑦ **ANALIZEAZĂ!** Un numărător invers de 4 biți se află în starea inițială 1001. Care va fi starea numărătorului după aplicarea a m impulsuri de intrare? Numărul m poate avea valorile 1, 4, 5, 8, 11, 17.
 ⑧ **PROGRAMEAZĂ!** Elaborați un program care simulează funcționarea unui numărător direct de n biți.

5.7. Generatoare de impulsuri

Impulsurile se utilizează în echipamentele numerice pentru a asigura funcționarea secvențială a acestora. De obicei, generatoarele de impulsuri se realizează pe baza porților logice și a elementelor de întârziere.

Elementul de întârziere reprezintă un circuit electronic care realizează funcția logică de repetare $y = x$, însă semnalul de ieșire y repetă semnalul de intrare x cu o întârziere de Δ unități de timp.

Din cursul de fizică e cunoscut faptul că viteza de propagare a semnalelor este finită. Prin urmare, orice conductor poate fi tratat ca un element de întârziere. Pentru a mări „inertitatea” circuitelor electronice și pentru a realiza întârzieri semnificative, în componența lor se includ condensatoare și rezistoare. În acest caz întârzierea Δ este determinată de capacitatea și rezistența componentelor respective.

În *figura 5.20* sunt prezentate simbolul și diagramele în timp ale elementului de întârziere.

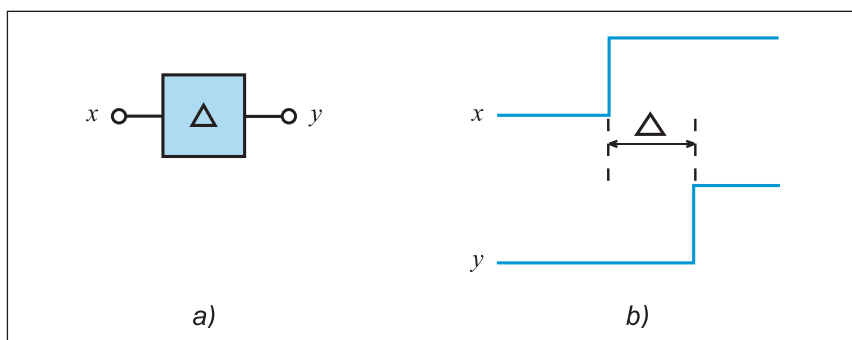


Fig. 5.20. Simbolul (a) și diagramele în timp (b) ale elementului de întârziere

Schema unui **generator de impulsuri periodice** realizat pe baza unui element de întârziere și a porții logice ȘI-NU este prezentată în figura 5.21. În starea inițială $x = 0$ și $y = 1$, iar la ieșirea elementului de întârziere se menține valoarea logică 1. Când la intrare se aplică semnalul de pornire $x = 1$, ieșirea devine egală cu 0 (fig. 5.22). În continuare, valoarea logică 0, după o întârziere Δ , este aplicată la a doua intrare a porții logice ȘI-NU. Astfel, ieșirea y devine egală cu 1. Valoarea logică 1, după o întârziere Δ , va fi din nou aplicată la intrarea porții logice ȘI-NU, impunând la ieșire valoarea $y = 0$ etc.

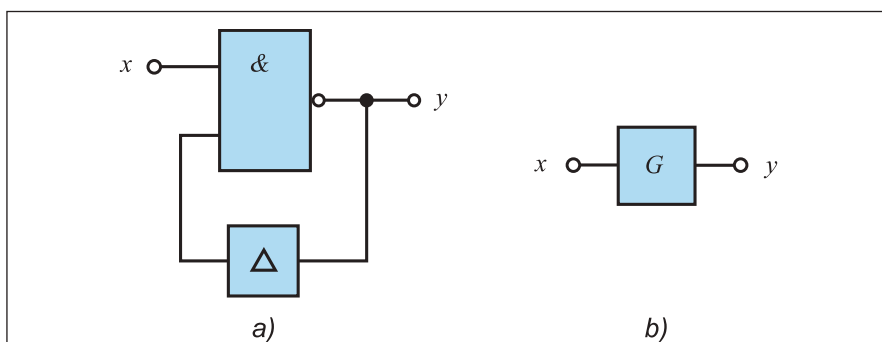


Fig. 5.21. Schema (a) și simbolul generatorului de impulsuri periodice (b)

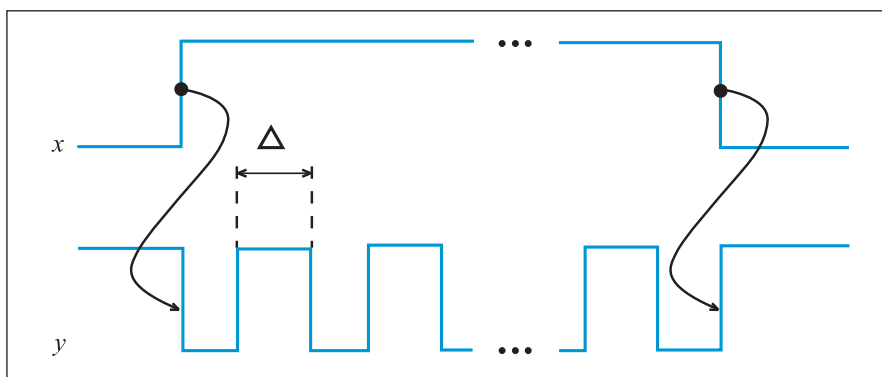


Fig. 5.22. Diagramele în timp ale generatorului de impulsuri periodice

Prin urmare, la ieșirea y a generatorului se va forma o succesiune de impulsuri cu durata Δ . Procesul de generare poate fi întrerupt prin aplicarea la intrare a semnalului de oprire $x = 0$.

Întrebări și exerciții

- ❶ Care este destinația elementului de întârziere? Desenați diagramele în timp ale elementului examinat.
- ❷ Explicați cum funcționează generatorul de impulsuri periodice. De ce depinde durata impulsurilor?
- ❸ PROIECTEAZĂ! Înlocuiți poarta logică *ȘI-NU* din componența generatorului de impulsuri periodice prezentat în *figura 5.21* printr-o poartă logică *SAU-NU*. Explicați cum va funcționa circuitul respectiv. Desenați diagramele în timp ale generatorului obținut.
- ❹ CERCETEAZĂ! E cunoscut faptul că variațiile mărimilor fizice nu pot avea loc instantaneu. Prin urmare, orice poartă logică are o întârziere δ , denumită **întârziere parazitară**, valoarea concretă a căreia depinde de particularitățile circuitului respectiv.
Excludeți din schema prezentată în *figura 5.21* elementul de întârziere, aplicând semnalul de la ieșirea porții logice *ȘI-NU* direct la intrarea ei. Explicați cum va funcționa circuitul obținut. De ce depinde durata impulsurilor la ieșirea porții logice? Desenați diagramele respective în timp.
- ❺ ANALIZEAZĂ! Circuitul secvențial constă dintr-o poartă logică *NU*, semnalul de ieșire al căreia este aplicat direct la intrarea ei. Cum va funcționa acest circuit?

Capitolul 6

STRUCTURA ȘI FUNCȚIONAREA CALCULATORULUI

6.1. Schema funcțională a calculatorului

Schema funcțională a calculatorului numeric este prezentată în *figura 6.1.*

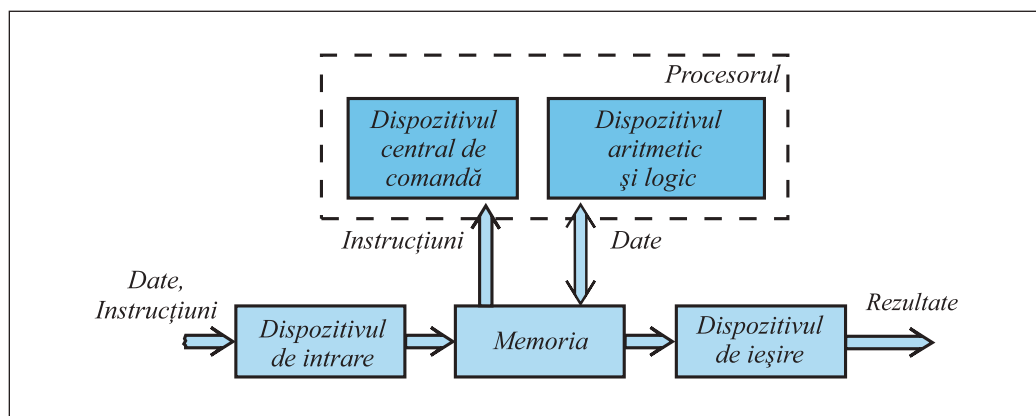


Fig. 6.1. Schema funcțională a calculatorului

Conform acestei scheme, calculatorul numeric conține următoarele **unități funcționale**:

- o unitate de memorie pentru a înmagazina datele inițiale, intermediare și finale ale problemei, precum și instrucțiunile care indică secvența calculelor;
- un dispozitiv aritmetic și logic necesar efectuării operațiilor aritmetice și logice elementare;
- unul sau mai multe dispozitive de intrare, respectiv, ieșire, necesare comunicării din exterior cu calculatorul;
- un dispozitiv central de comandă și control care generează o succesiune de semnale de comandă necesare executării secvențiale a instrucțiunilor.

Dispozitivul aritmetic și logic și dispozitivul central de comandă formează **unitatea centrală de prelucrare a informației** sau, mai pe scurt, **procesorul**.

Memoria calculatoarelor moderne este organizată în două niveluri, și anume: o unitate de memorie internă cu o viteză mare de lucru și una sau mai multe unități de memorie externă cu o viteză mai redusă, însă cu o capacitate mult mai mare decât cea a memoriei interne.

Memoria internă (numită uneori memorie principală, centrală sau operativă) păstrează programul în curs de executare și datele folosite de acesta, prezența ei fiind o condiție esențială pentru funcționarea calculatorului.

Memoria externă are rolul de a păstra cantități mari de informație și programe folosite frecvent pentru a putea fi aduse într-un interval de timp mic în memoria internă. În prezent ca memorii externe sunt utilizate unitățile cu discuri sau benzi magnetice, unitățile cu discuri optice etc.

Unitățile de memorie externă și dispozitivele de intrare-ieșire sunt numite **echipamente periferice**.

Pentru a asigura o interacțiune eficientă a procesorului, a memoriei interne și a echipamentelor periferice, în cazul calculatoarelor personale schema funcțională se realizează fizic conform schemei-bloc prezentate în *figura 6.2*.

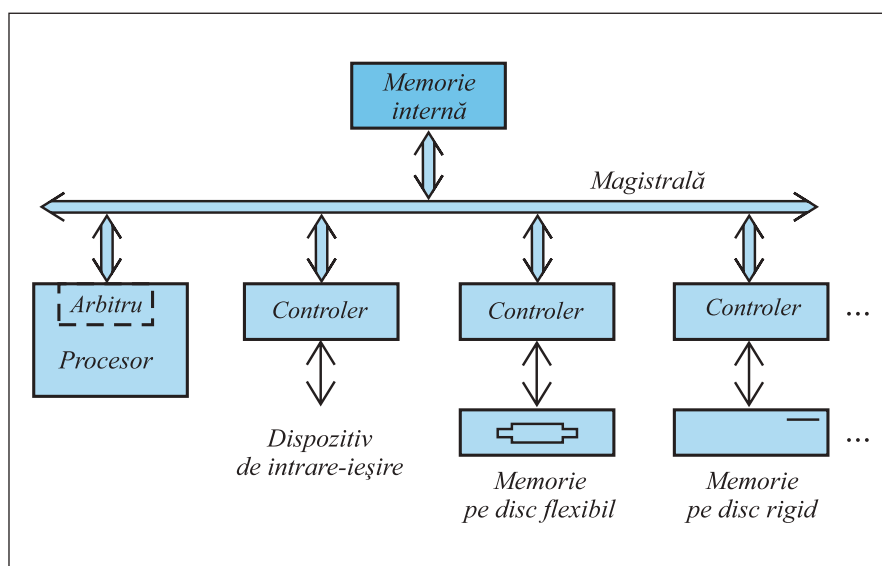


Fig. 6.2. Schema-bloc a unui calculator personal

Din analiza *figurii 6.2* rezultă că toate calculatoarele moderne au o **configurație modulară**. Fiecare modül, și anume controlerul, imprimantele, unitățile de disc magnetic etc., funcționează și, în consecință, pot fi incluse sau excluse din componența calculatorului independent unul de altul. Prin urmare, configurația calculatorului poate fi modificată în funcție de destinația sistemului de calcul.

De exemplu, un sistem editorial va include mai multe tipuri de imprimante: mecanice pentru textele în curs de prelucrare, laser sau color pentru paginile machetate deja, cititoare de desene și fotografii (scanere) etc. Un sistem destinat gestionării rapide a unui volum mare de date va include mai multe discuri magnetice, iar un calculator utilizat pentru montarea filmelor video va fi dotat cu camere de luat vederi și vizualizatoare de o rezoluție adecvată, cu tastaturi similare pupitrului regizoral etc.

Întrebări și exerciții

- ❶ Numiți unitățile funcționale ale calculatorului și explicați destinația lor.
- ❷ Care este rolul memoriei interne? Cum se realizează memoria externă a calculatoarelor moderne?
- ❸ Numiți echipamentele periferice pe care le cunoașteți dvs.
- ❹ Numiți componentele unui calculator personal și explicați destinația lor.
- ❺ Care este structura și cum interacționează componentele unui calculator?
- ❻ Cum poate fi modificată configurația unui sistem de calcul? Care sunt avantajele configurației modulare a calculatorului?
- ❼ CERCETEAZĂ! Desenați schema-bloc a calculatorului cu care lucrați dvs. Care componente sunt obligatorii și care opționale pentru funcționarea calculatorului?
- ❽ EXPERIMENTEAZĂ! Cum poate fi conectată la un calculator personal o unitate suplimentară de disc magnetic? O imprimantă laser? Un cititor de documente? O cameră de luat vederi?
- ❾ EXPLOREAZĂ! Utilizând un motor de căutare, colectați din Internet informațiile referitoare la evoluția schemelor funcționale ale calculatoarelor digitale și scrieți un mic eseu despre acest subiect. Puneți în evidență factorii ce au dus la utilizarea pe scară largă a schemelor funcționale cu magistrală.

6.2. Formatul instrucțiunilor

Pentru a rezolva o problemă, calculatorul trebuie să cunoască în fiecare moment atât operația pe care urmează să o execute, cât și datele care participă în operație. Aceste operații sunt comunicate calculatorului prin intermediul instrucțiunilor.

Instrucțiunea calculatorului reprezintă o succesiune de cifre binare prin care se indică procesorului operația de executat și amplasamentul (locul) operanzilor.

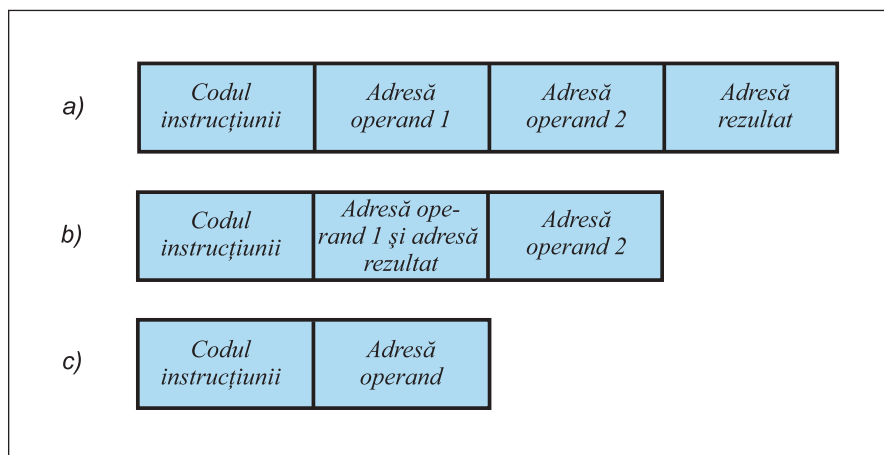


Fig. 6.3. Formatul instrucțiunilor cu trei (a), două (b) și o singură adresă (c)

Sucesiunea binară respectivă, denumită uneori și **cuvânt instrucțiune**, este împărțită în câmpuri, fiecare câmp având o semnificație precisă. Numărul și semnificația câmpurilor poartă denumirea de **formatul instrucțiunii**. În *figura 6.3*, p. 165 sunt prezentate formatele utilizate în calculatoarele moderne.

În general, pentru executarea unei operații este necesar ca instrucțiunea să conțină trei adrese (*fig. 6.3a*). Primele două adrese sunt folosite pentru obținerea celor doi operanzi asupra cărora se va efectua operația specificată de câmpul *Codul instrucțiunii*. Rezultatul operației va fi depus pe adresa specificată de câmpul *Adresă rezultat*.

Să analizăm un exemplu. Presupunem că operațiile aritmetice și logice sunt codificate după cum urmează (pentru simplitate vom utiliza echivalentele zecimale ale câmpurilor binare respective):

01 – adunarea;

02 – scăderea;

03 – operația logică *ȘI*;

04 – operația logică *SAU*.

Instrucțiunea

01 100 110 215

va impune procesorul să adune numerele din locațiile 100 și 110 și să depună suma obținută în locația 215.

Instrucțiunea

02 100 110 215

comunică procesorului că din numărul înscris în locația 100 se scade numărul din locația 110. Rezultatul obținut va fi depus în locația 215.

În mod similar, instrucțiunea

03 200 300 100

specifică operația logică *ȘI* asupra biților cuvintelor din locațiile 200 și 300. Rezultatul operației va fi depus în locația 100.

Se observă că într-o instrucțiune nu se specifică **valoarea operanzilor**, ci **adresele locațiilor** în care pot fi găsiți operanzii respectivi. Acest fapt permite utilizarea unuia și aceluiași program pentru prelucrarea oricăror date inițiale. Faptul că instrucțiunile lucrează cu adrese al căror conținut trebuie prelucrat, și nu cu însuși conținutul, constituie un principiu fundamental al calculatoarelor numerice, care permite ca un program să fie elaborat și introdus în calculator independent de datele concrete asupra cărora se aplică.

În formatul cu trei adrese (*fig. 6.3a*), adresele sunt **specificate explicit**. Pentru o reprezentare mai compactă a instrucțiunilor se utilizează **specificarea implicită** a unor adrese. În acest caz, cuvântul instrucțiune nu conține un câmp special pentru adresa implicită.

Dacă rezultatul obținut în urma executării unei instrucțiuni se depune pe adresa unuia dintre operanzi, formatul respectiv va avea numai două adrese (*fig. 6.3b*). Prin

urmare, adresa rezultatului este specificată implicit. De exemplu, instrucțiunea

01 100 110

va impune procesorul să adune numerele din locațiile 100 și 110 și să depună suma obținută în locația 100. Evident, după înscrierea sumei, numărul inițial din locația 100 va fi suprimat.

S-a constatat că formatul cu două adrese, în prezent cel mai răspândit, permite scrierea de programe având un număr de instrucțiuni comparabil cu cel obținut atunci când s-ar utiliza mai multe adrese.

Formatul cu o singură adresă (*fig. 6.3c*) se utilizează în calculatoarele procesorului cărora include un registru special, denumit **acumulator**. În acumulator se păstrează primul operand și se depune rezultatul executării operației respective. Prin urmare, adresa primului operand și adresa rezultatului sunt specificate implicit. De exemplu, instrucțiunea cu o singură adresă

01 100

va aduna numărul din acumulator cu numărul din locația 100, iar suma obținută va fi depusă în acumulator. Respectiv, numărul inițial din acumulator va fi suprimat.

Instrucțiunile cu o singură adresă sunt eficiente din punctul de vedere al lungimii cuvântului și al rapidității calculatorului. Totuși un program scris cu instrucțiuni având o singură adresă va conține mai multe instrucțiuni decât în cazul în care se folosesc instrucțiuni cu două sau trei adrese.

Menționăm că toate calculatoarele moderne pot avea instrucțiuni de diferite formate. Informația referitoare la formatul fiecărei instrucțiuni se indică în câmpul *Codul instrucțiunii*.

Întrebări și exerciții

- ❶ Enumerați formatele instrucțiunilor utilizate în calculatoarele moderne. Explicați modul de specificare implicită a unor adrese.
- ❷ Explicați semnificația câmpurilor instrucțiunilor cu trei sau două adrese.
- ❸ Cum se specifică adresele operanzilor și ale rezultatului în cazul instrucțiunilor cu o singură adresă?
- ❹ Explicați cum vor fi executate următoarele instrucțiuni cu trei adrese:

a) 01 200 201 202;

c) 03 100 150 250;

b) 04 202 201 200;

d) 02 250 300 310.

Operațiile aritmetice și logice sunt codificate ca și în exemplul din paragraful de față.

- ❺ Care va fi conținutul locației 100 după executarea instrucțiunii

01 200 300 100,

dacă în locațiile 200 și 300 sunt înscrise numerele 17 și, respectiv, 31?

⑥ Explicați cum vor fi executate următoarele instrucțiuni cu două adrese:

a) 01 200 201;

c) 03 100 150;

b) 04 202 201;

d) 02 250 300.

⑦ Care va fi conținutul locației 200 după executarea instrucțiunii

01 200 100,

dacă în locațiile 100 și 200 sunt înscrise numerele 18 și, respectiv, 32?

⑧ Explicați cum vor fi executate următoarele instrucțiuni cu o singură adresă:

a) 01 100;

c) 02 400;

b) 03 200;

d) 04 150.

⑨ Care va fi conținutul acumulatorului după executarea instrucțiunii

01 100,

dacă inițial în locația 100 era înscris numărul 12, iar în acumulator numărul 26?

⑩ Care sunt avantajele și dezavantajele formatelor cu trei, două sau cu o singură adresă?

⑪ **STUDIU DE CAZ.** Găsește pe Internet informații referitoare la tipurile de calculatoare ce utilizează formate cu trei, două sau cu o singură adresă. Identificați domeniile de utilizare, avantajele și neajunsurile tipurilor respective de calculatoare.

⑫ **EXPLOREAZĂ!** Utilizând un motor de căutare, colectați din Internet informațiile referitoare la evoluția formatelor instrucțiunilor utilizate în calculatoarele digitale și scrieți un mic eseu despre acest subiect. Puneți în evidență factorii ce au dus la utilizarea pe scară largă a formatelor utilizate în calculatoarele personale.

6.3. Tipuri de instrucțiuni

Instrucțiunile unui calculator se împart în patru grupe:

- instrucțiuni operaționale, care efectuează operații aritmetice și logice asupra datelor specificate prin operanzi;
- instrucțiuni de transfer, care deplasează informația între registre și/sau locații fără a modifica informația transferată;
- instrucțiuni de salt, care în urma verificării unor condiții modifică analiza și execuția secvențială a instrucțiunilor din program;
- instrucțiuni de intrare-ieșire care permit comunicarea calculatorului cu exteriorul.

Instrucțiunile operaționale prelucreză datele păstrate în locațiile memoriei interne și în registrele procesorului. Cele mai cunoscute instrucțiuni ale acestei grupe sunt cele care efectuează operațiile aritmetice de bază: *adunarea*, *scăderea*, *înmulțirea* și *împărțirea*.

Instrucțiunile logice de tipul *ȘI*, *SAU*, *NU* sunt instrucțiuni operaționale care acționează asupra pozițiilor individuale ale informației binare. În categoria instrucțiunilor operaționale întâlnim și instrucțiuni de tipul: *șterge* conținutul unei locații sau al unui registru, *complementează* conținutul unei locații, *crește* cu o unitate conținutul unui registru etc. În fine, în categoria instrucțiunilor operaționale sunt incluse instrucțiunile de *deplasare* a informației, în care partea de adresă a instrucțiunii conține un număr întreg, ce specifică numărul pozițiilor cu care se face deplasarea.

Instrucțiunile de transfer deplasează informația dintre locațiile memoriei interne, între registre sau între locații și registre fără a altera conținutul informației transferate. Instrucțiunea trebuie să specifice explicit sau implicit adresa-sursă și adresa de destinație a transferului. În timpul transferului și după transfer, informația din sursă rămâne neschimbată. Cele mai uzuale instrucțiuni ale acestei grupe sunt cele prin care conținutul unei locații trece într-un anumit registru, registrul acumulator, precum și instrucțiunea de transfer invers: dintr-un registru într-o locație a memoriei interne.

Instrucțiunile de salt se utilizează pentru modificarea ordinii de execuție a instrucțiunilor. În mod normal, instrucțiunile unui program sunt analizate și executate în mod secvențial, în ordinea în care sunt așezate în memorie. Această ordine poate fi schimbată cu ajutorul instrucțiunilor de salt condiționat sau necondiționat.

Instrucțiunile de salt condiționat permit alegerea continuării programului pe o anumită ramură, în funcție de o condiție de test realizată. Folosirea instrucțiunilor de salt condiționat dau posibilitate utilizatorului să introducă decizii logice în procesul execuției programului.

O instrucțiune de salt necondiționat conține, în partea de adresă, adresa instrucțiunii care va fi executată în continuare.

Instrucțiunile de intrare-ieșire permit comunicarea calculatorului cu echipamentele periferice. Echipamentul cu care se va efectua operația de intrare-ieșire se specifică în partea de adresă a instrucțiunii. De regulă, instrucțiunile de acest tip conțin atât informații legate de natura schimbului de date, adică introducerea sau extragerea lor, cât și comenzi necesare funcționării corecte a periferiei. Tot în aceste instrucțiuni se specifică și registrele sau locațiile în care vor fi depuse sau din care vor fi luate datele respective.

Întrebări și exerciții

- 1 Cum se clasifică instrucțiunile unui calculator? Care este destinația instrucțiunilor din fiecare grupă?
- 2 Dați câteva exemple de instrucțiuni operaționale. Estimați numărul instrucțiunilor operaționale posibile.
- 3 Care este destinația instrucțiunilor de transfer? Estimați numărul instrucțiunilor posibile de transfer.
- 4 Când și cum se utilizează instrucțiunile de salt? Care condiții de test pot fi analizate de aceste instrucțiuni?

- ⑤ Care este destinația instrucțiunilor de intrare-ieșire? Ce informații conțin aceste instrucțiuni?
- ⑥ EXPLOREAZĂ! Aflați clasificarea și componența instrucțiunilor calculatoarelor cu care lucrați dvs.

6.4. Limbajul cod-calculator și limbajul de asamblare

Pentru a rezolva o problemă, în memoria calculatorului trebuie să fie încărcate programul respectiv și datele de prelucrat. Instrucțiunile programului și datele de prelucrat se înmagazinează în memoria internă sub forma unor succesiuni de cifre binare pe care dispozitivul central de comandă le poate extrage și interpreta.

Programele reprezentate în formă de succesiuni binare direct executabile de calculator se numesc programe în limbaj cod-calculator sau programe în limbaj mașină.

Pentru utilizator programul în cod-calculator poate fi prezentat în formă de șiruri de cifre binare sau, mai compact, de cifre octale, zecimale sau hexazecimale organizate pe locații ale memoriei.

Elaborarea programelor în limbaj cod-calculator este un lucru extenuant și ineficient. Pentru a simplifica procesul de elaborare a programelor, s-a convenit ca instrucțiunile să fie scrise într-un limbaj simbolic, denumit **limbaj de asamblare**. În acest limbaj codurile instrucțiunilor se reprezintă printr-un grup de caractere astfel ales, încât să sugereze cât mai bine natura operației. Acest grup de caractere, de regulă trei, este cunoscut sub numele **mnemonica instrucțiunii**.

De exemplu, codurile instrucțiunilor calculatorului ipotetic din paragraful precedent pot fi notate simbolic conform *tabelului 6.1*.

Tabelul 6.1

Mnemonica instrucțiunilor

<i>Cod instrucțiune</i>	<i>Mnemonica</i>	<i>Semnificația</i>
01	INC	Încarcă acumulatorul
02	MEM	Memorează acumulatorul
03	ADU	Adunare
04	SCD	Scădere
05	SLT	Salt necondiționat
06	SLTC	Salt condiționat
07	STP	Stop

Adresele locațiilor memoriei interne pot fi specificate prin denumiri simbolice alese de utilizator, denumiri care sugerează semnificația conținutului locațiilor respective.

De exemplu, locația 185, în care se va păstra numărul x , poate fi notată prin X , locația 213 pentru numărul y se va nota prin Y , locația 200, în care va fi depusă

suma $x + y$, se va nota prin S . În limbajul de asamblare fragmentul de program pentru adunarea numerelor x și y va avea forma:

```
INC X
ADU Y
MEM S.
```

În general, există o corespondență directă între scrierea instrucțiunii în limbajul de asamblare și scrierea în limbajul cod-calculator, ceea ce face ușoară traducerea (traducerea) limbajului de asamblare în limbajul cod-calculator.

Traducerea constă în înlocuirea mnemonicii instrucțiunii și a adreselor simbolice prin șirurile binare respective. Această înlocuire este făcută de un program special, denumit program de asamblare sau asamblor.

Limbajele cod-calculator și de asamblare sunt **limbaje dependente de calculator**. Această dependență constă în faptul că formatul, codurile și mnemonica instrucțiunilor exprimă structura internă a calculatorului. Programele elaborate în aceste limbaje sunt cele mai scurte și rapide, însă procesul de programare necesită un mare volum de muncă. Simplificarea procesului de programare se asigură prin utilizarea limbajelor independente de calculator (*FORTRAN*, *BASIC*, *PASCAL*, *C* etc.), în care operațiile de prelucrare și tipurile de date nu sunt legate de echipamentele calculatorului. Însă, cu regret, detașarea utilizatorului de structura internă a calculatorului diminuează eficacitatea programelor respective.

Întrebări și exerciții

- 1 Care este diferența dintre limbajul cod-calculator și limbajul de asamblare?
- 2 Cum se exprimă codurile instrucțiunilor și adresele locațiilor într-un limbaj de asamblare?
- 3 Care este destinația și cum se realizează traducerea programelor scrise într-un limbaj de asamblare?
- 4 **PROGRAMEAZĂ!** Se consideră că denumirea simbolică X semnifică locația 100, denumirea Y – locația 101, iar denumirea S – locația 102. Exprimați în limbajul de asamblare (*tab. 6.1*) următoarele programe:

a) 01 100
04 101
02 102
02 100

b) 01 100
02 100
03 100

c) 01 101
04 100
02 102

d) 01 101
03 101
03 101
03 101
02 102

e) 01 100
02 101
03 101
02 100

f) 01 100
02 102
01 101
02 100

Care va fi conținutul locațiilor 100, 101 și 102 până și după execuția fiecărui program?

- ⑤ Se consideră că denumirile simbolice X , Y și S specifică, respectiv, locațiile 100, 200 și 300. Translați următoarele programe scrise în limbajul de asamblare:

a)	INC X SCD Y MEM S INC Y	b)	INC X INC X SLTC S STP	c)	INC Y ADU X MEM S STP
d)	INC Y ADU Y ADU Y ADU Y MEM S	e)	INC X MEM S INC Y MEM X INC X	f)	INC X MEM S INC Y MEM X INC S

Explicați cum va fi executat fiecare program.

- ⑥ **STUDIUL DE CAZ.** Care este diferența dintre limbajele dependente și limbajele independente de calculator? Care sunt avantajele și dezavantajele fiecărui limbaj?
- ⑦ **EXPLOREAZĂ!** Utilizând Internetul în calitate de sursă de informare, alcătuiți o scurtă caracteristică a limbajelor cod-calculator și a celor de asamblare ale calculatoarelor cu care lucrați: codurile și instrucțiunile, clasificarea acestora.

6.5. Resursele tehnice și resursele programate ale calculatorului

Numărul total al instrucțiunilor unui calculator depinde, în primul rând, de capacitatea lui. În cazul unui sistem mare de calcul, acest număr poate depăși 1 000, pe când la calculatoarele foarte mici el nu este mai mare de 100. O anumită operație poate fi efectuată la unele calculatoare cu o singură instrucțiune, pe când aceeași operație, în alte calculatoare, pentru care nu există o instrucțiune specifică, este efectuată cu ajutorul unei succesiuni de alte instrucțiuni existente.

Operațiile efectuate intern de componentele electronice ale calculatorului sunt cunoscute ca **operații implementate prin echipamente**, pe când operațiile efectuate cu ajutorul unei secvențe de instrucțiuni sunt cunoscute ca **operații implementate prin program**. De exemplu, operația de extragere a rădăcinii pătrate într-un anumit calculator poate fi efectuată prin echipament electronic, în alt calculator – printr-un subprogram. Delimitarea între implementarea prin echipamente și program depinde de calculator. În *figura 6.4* este prezentată o astfel de delimitare în cazul unui calculator cu posibilități medii.

Operațiile implementate prin echipamente se efectuează prin execuția unei singure instrucțiuni, pe când o operație implementată prin program necesită execuția mai multor instrucțiuni. Prin urmare, operațiile implementate prin echipamente se efectuează mai rapid, însă calculatorul respectiv este mai complex și, în consecință, mai scump. Din contra, operațiile implementate prin program se efectuează mai lent, însă calculatoarele respective sunt mai simple și, evident, mai ieftine.

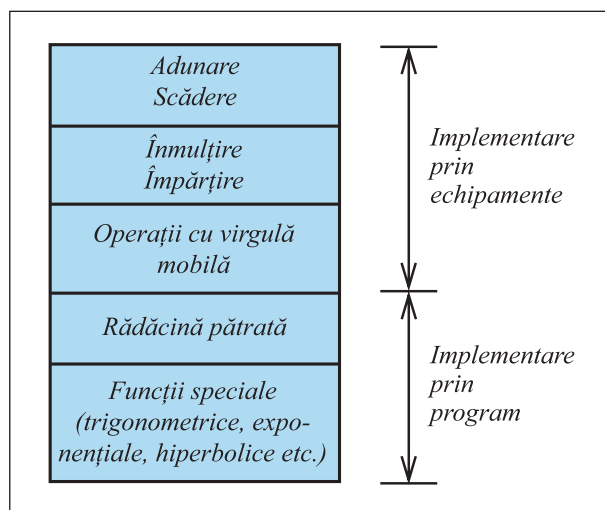


Fig. 6.4. Delimitarea între implementarea prin echipamente și prin program

Din analiza principiilor de funcționare a procesorului rezultă că toate echipamentele unui calculator devin inutile în absența programelor care guvernează efectuarea operațiilor necesare pentru rezolvarea unei probleme. Exact în același mod, programele sunt fără niciun folos în absența echipamentelor numerice respective. Prin urmare, utilizarea tehnicii de calcul este posibilă numai în prezența atât a echipamentelor, denumite **resurse tehnice**, cât și a programelor respective, denumite **resurse programate**.

Resursele tehnice ale unui sistem de calcul modern includ procesorul, memoria internă, unitățile de memorie externă, echipamentele de intrare-ieșire etc. **Resursele programate** vor include subprogramele care realizează operațiile implementate prin program, programele care simplifică accesul la unitățile de intrare-ieșire, asambloarele, editoarele de texte, compilatoarele limbajelor algoritmice și, evident, programele elaborate de fiecare utilizator.

Menționăm că în literatura de specialitate resursele tehnice uneori sunt denumite prin cuvântul englez *hardware* („produse de metal”), iar resursele programate – prin cuvântul *software* („produse moi”). Respectiv, implementarea prin echipamente se numește implementare prin *hardware*, iar implementarea prin program – implementare prin *software*.

Întrebări și exerciții

- ❶ De ce depinde numărul total al instrucțiunilor unui calculator?
- ❷ Cum se efectuează operațiile în cazurile implementării prin echipamente și în cele ale implementării prin program?
- ❸ EXPLOREAZĂ! Analizând repertoriul instrucțiunilor calculatorului la care lucrați dvs., determinați cum sunt implementate următoarele operații:
 - înmulțirea și împărțirea numerelor binare;
 - adunarea și scăderea numerelor cu virgulă mobilă;

- înmulțirea și împărțirea numerelor cu virgulă mobilă;
 - extragerea rădăcinii pătrate;
 - calcularea funcțiilor trigonometrice.
- 4 STUDIU DE CAZ. Care sunt avantajele și dezavantajele implementării prin echipamente? Dar ale implementării prin program?
- 5 EXPERIMENTEAZĂ! Numiți resursele tehnice și resursele programate ale unui sistem de calcul. Care sunt resursele respective în cazul calculatorului la care lucrați dvs.?

6.6. Memorii externe pe benzi și discuri magnetice

Principiul de funcționare a memoriilor examinate constă în înregistrarea informației pe un strat magnetic aflat în mișcare. Stratul magnetic este depus pe un suport neutru, acesta fiind, de regulă, o bandă de material plastic sau un disc de aluminiu. În calitate de strat magnetic se folosește mai frecvent oxidul de fier sau pelicule metalice subțiri de cobalt-nichel cu depunere în vid.

Înregistrarea sau citirea informației se efectuează cu ajutorul unui cap magnetic reprezentat în *figura 6.5*.

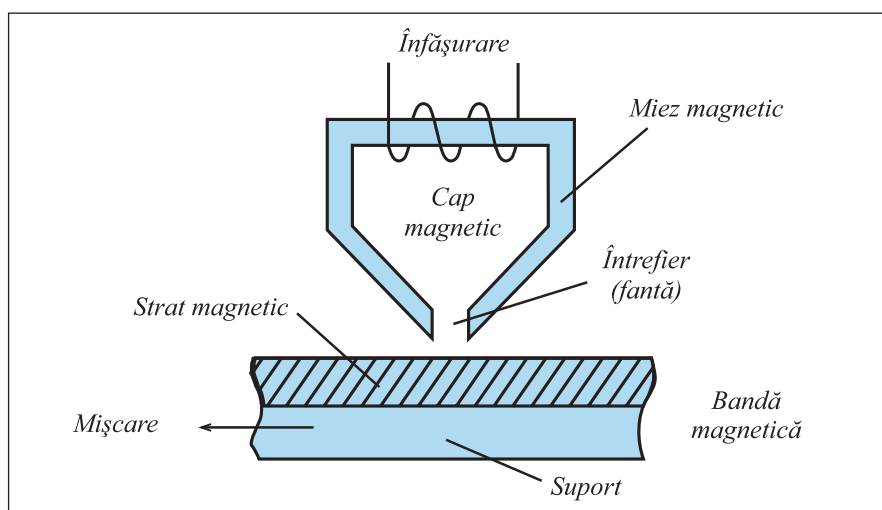


Fig. 6.5. Cap magnetic pentru înscrierea și citirea informației

Capul constă dintr-un miez, de regulă, din tole (foițe) de permalloy foarte subțiri (0,05 mm) și o înfășurare.

Într-un strat magnetic neînregistrat, câmpurile magnetice ale particulelor de oxid de fier sunt orientate haotic, anulându-se reciproc. Pentru a **înregistra** cifra binară 0 sau 1, prin înfășurarea capului magnetic se trece impulsul respectiv de curent. Impulsul care traversează înfășurarea creează în întrefier un câmp magnetic intens care magnetizează stratul aflat în momentul actual sub cap. Direcția de magnetizare, deci informația binară înscrisă, depinde de direcția

curentului în înfășurarea capului magnetic. În figura 6.6 este prezentat un exemplu de înregistrare a informației binare 101101 pe un strat magnetic aflat în mișcare.

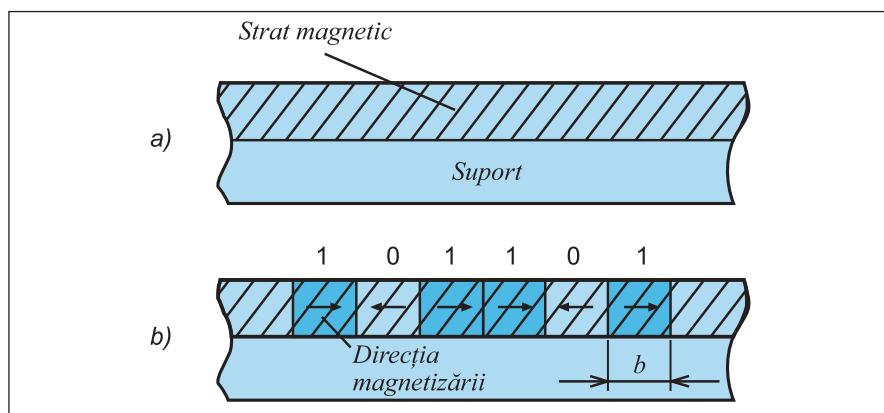


Fig. 6.6. Starea stratului magnetic până (a) și după înregistrare (b)

Distanța b determină lungimea porțiunii necesare pentru a memora o cifră binară. Valoarea lui b depinde de viteza de mișcare a suportului, de proprietățile fizice ale stratului magnetic, de elementele constructive ale capului magnetic etc.

Numărul de elemente de memorie binară pe unitatea de lungime a suportului se numește densitatea de înregistrare a informației.

În cazul înregistrărilor magnetice, densitatea este dată de mărimea $1/b$. Valorile practice variază după tipul dispozitivului de memorare și firma constructoare, fiind de ordinul sutelor și miilor de biți pe milimetru de lungime a suportului.

În timpul operației de **citire** câmpul magnetic al particulelor de oxid de fier, trecând prin dreptul întrefierului (fig. 6.5), induce în înfășurarea capului magnetic un semnal de ordinul 10^{-3} volți. Acest semnal este amplificat și transformat în semnal-standard care reprezintă cifra binară respectivă 0 sau 1.

În marea lor majoritate **memoriile externe pe bandă magnetică** reprezintă echipamente periferice autonome care transferă informația spre/de la memoria internă după receptarea comenzilor corespunzătoare de la procesorul calculatorului. O unitate de memorie pe bandă magnetică (fig. 6.7, p. 176) include mecanismul de antrenare a benzii, dispozitivul de scriere-citire și circuitele de comandă aferente.

O operație de citire sau scriere se realizează în timpul deplasării benzii. Între două operații succesive banda este oprită. Evident, informațiile înregistrate pot fi citite numai în ordinea amplasării lor fizice pe bandă. Datorită acestui fapt, unitățile de bandă magnetică sunt denumite unități de memorie externă cu **acces secvențial**.

Timul necesar pentru selectarea unei informații din multitudinea datelor memorate pe un suport se numește timp de acces.

Timul de acces al unității de bandă magnetică depinde de viteza benzii și de locul amplasării informației de citit: la începutul, la sfârșitul sau la mijlocul benzii. În cazul informațiilor înregistrate la sfârșitul benzii, timpul de acces este de ordinul minutelor.

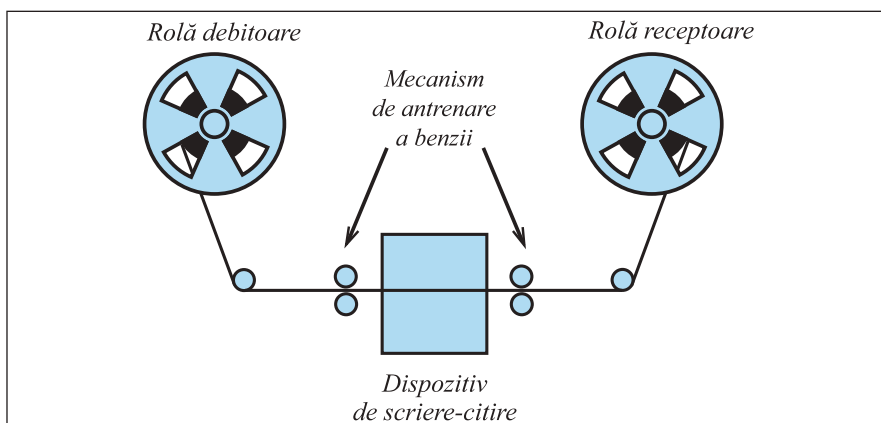


Fig. 6.7. Unitate de bandă magnetică

Capacitatea de memorare a unei benzi magnetice depinde de densitatea înregistrării, numărul de piste, lungimea benzii și are valori de ordinul 10^8 octeți.

Din cauza timpului mare de acces și capacitatea de memorie relativ mică, benzile magnetice se utilizează, în general, numai pentru arhivarea informației pe perioade mari de timp, până la 50 de ani.

Unitatea de discuri magnetice reprezintă la etapa actuală cea mai răspândită memorie externă a calculatoarelor numerice. Suportul informației este format dintr-un pachet de discuri, care poate fi fix sau amovibil, rotit cu o viteză de ordinul miilor de rotații pe minut. Discurile sunt acoperite cu un strat de material feromagnetic. Informația este înregistrată pe ambele fețe de-a lungul unor piste concentrice. Pentru aceasta fiecare față este explorată în plan orizontal de un cap magnetic mobil (fig. 6.8).

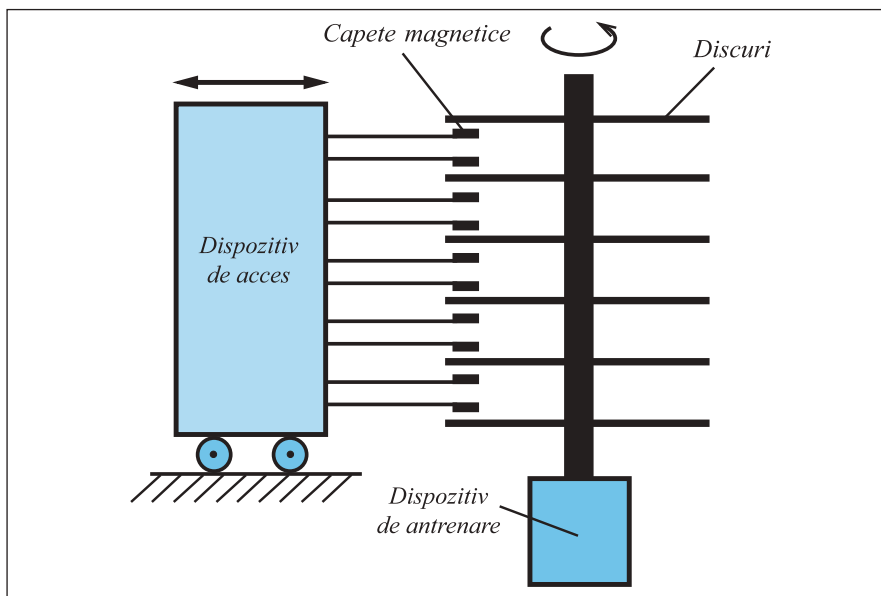


Fig. 6.8. Unitate de discuri cu capete mobile

Capetele sunt montate pe un braț extensibil și acționate de un mecanism pentru a ajunge în dreptul pistei selectate. Toate capetele unei unități de memorie cu discuri sunt poziționate simultan.

Timpul de acces al unității de discuri se compune din timpul necesar deplasării ansamblului de capete magnetice de pe cilindrul curent pe cilindrul indicat și din timpul necesar ca sectorul respectiv al discului să ajungă în dreptul capului magnetic. În practică se utilizează **timpul mediu de acces**, care pentru unitățile de disc moderne este de ordinul 10^{-3} secunde.

Amintim că în calculatoarele de performanță se utilizează unități de discuri cu capete magnetice fixe – câte un cap magnetic pentru fiecare pistă. Aceste unități asigură un timp de acces de ordinul 10^{-4} secunde, însă sunt foarte scumpe.

În trecut, pentru schimbul de informații între calculatoare se utilizau discuri singulare din material plastic, denumite **discuri flexibile** sau **dischete**. Aceste discuri erau încorporate într-o casetă din plastic sau în plicuri speciale. Organizarea fizică a informației era aceeași ca și în cazul pachetelor de discuri, însă unitățile respective erau mult mai simple și, evident, mai ieftine. Pentru a le deosebi de discurile flexibile, discurile convenționale ale calculatoarelor personale au fost denumite **discuri rigide**, **hard diskuri** sau **wincestere**.

Capacitatea de memorare a discurilor magnetice depinde de numărul de suprafețe ale pachetului, numărul de cilindri și densitatea de înregistrare. La momentul actual este obținută capacitatea de ordinul 10^{12} octeți pentru un pachet de discuri.

Menționăm faptul că în ultimii ani, în locul sau suplimentar la discurile rigide au început să fie folosite dispozitive de stocare a datelor care folosește memorii cu semiconductori, denumite unități cu cipuri sau SSD-uri (Solid-State Drive). SSD-urile sunt mai rezistente la șocurile mecanice, având timpul de acces mai scăzut. La moment, capacitatea de memorare a acestor unități este mai mică decât a discurilor rigide, iar costul lor este mai mare.

Întrebări și exerciții

- 1 Cum sunt reprezentate cifrele binare 0 și 1 în înregistrările magnetice?
- 2 Care este destinația capului magnetic?
- 3 De ce depinde densitatea de înregistrare magnetică a informației?
- 4 Cum se citește informația înregistrată pe un strat magnetic?
- 5 Explicați cum funcționează unitatea de memorie pe bandă magnetică din *figura 6.7*.
- 6 CERCETEază! De ce depinde capacitatea de memorare a unei benzi magnetice?
- 7 O bandă magnetică are lungimea de 750 m. Înregistrarea informației se efectuează pe 8 piste plus pista bitului de paritate. Capacitatea de memorare a benzii este de 47 *Megaocteți*. Determinați densitatea de înregistrare.
- 8 Viteza benzii magnetice este 2 m/s. În rola debitoare (*fig. 6.7*) sunt 750 m de bandă. Determinați timpul de acces la informația de la mijlocul benzii.
- 9 Cum funcționează o unitate de discuri cu capete mobile?
- 10 Cum este organizată informația pe un pachet de discuri magnetice?

- ❶ Care este diferența dintre memoriile externe cu acces direct și cu acces secvențial?
- ❷ De ce depinde timpul de acces al unității de discuri magnetice?
- ❸ CREEAZĂ! Utilizând sursele de informare din Internet, scrieți un mic eseu despre evoluția discurilor flexibile.
- ❹ EXPERIMENTEAZĂ! Pentru unitatea de disc rigid cu care lucrați dvs. determinați:
 - capacitatea discului;
 - timpul mediu de acces.
- ❺ CREEAZĂ! Utilizând sursele de informare din Internet, scrieți un mic eseu despre evoluția discurilor rigide.
- ❻ STUDIU DE CAZ. Elaborați un mic studiu care pune în evidență avantajele și neajunsurile discurilor rigide și ale memoriilor cu semiconductori (SSD-uri). Analizați frecvențele de utilizare a acestor unități de stocare a datelor în calculatoarele personale de tip desktop și portabile (notebook-uri/laptop-uri) comercializate în țara noastră.

6.7. Memorii externe pe discuri optice

Principiul de funcționare a **memoriilor pe discuri optice** constă în înregistrarea informației pe un strat reflectorizant aflat în mișcare. Stratul reflectorizant din aluminiu, aur sau argint este depus pe un suport transparent din masă plastică.

În funcție de modul de scriere și citire a informației, deosebim:

1) Discuri optice **numai pentru citire**. Informația pe astfel de discuri se înscrie de fabricant și nu poate fi modificată de utilizator. Abrevierea engleză a acestor discuri este *CD-ROM (Compact Disc-Read Only Memory)*.

2) Discuri optice **inscriptibile**. Informația pe astfel de discuri se înscrie de utilizator o singură dată, în continuare discul fiind disponibil numai pentru citire. Abrevierea engleză a acestor discuri este *CD-R (Compact Disc-Recordable)*.

3) Discuri optice **reinscriptibile**. Discurile în cauză permit mai multe cicluri de scriere/ștergere a informației. Abrevierea acestor discuri este *CD-RW (Compact Disc-ReWritable)*.

Pentru a asigura compatibilitatea unităților de citire, formatul datelor și dimensiunile discurilor optice sunt standardizate. În *figura 6.9* este reprezentată structura discului optic *numai pentru citire* destinat publicului larg.

Înregistrarea cifrelor binare pe astfel de discuri constă dintr-o succesiune de adâncituri (în limba engleză *pit*) realizate pe una dintre suprafețele discului. Aceste adâncituri sunt despărțite de mici pauze și sunt plasate pe suprafața discului sub forma unei piste în spirală.

Dimensiunile adânciturilor sunt de ordinul unui micron ($1 \text{ micron} = 10^{-3} \text{ milimetri}$), distanța dintre spire este de 1,6 microni, lungimea spiralei fiind de 5 300 m. Discul conține 20 000 de piste (spire) pe care se află circa $6 \cdot 10^9$ adâncituri. Capacitatea de memorare a discului este de 640 *Megaocteți*.

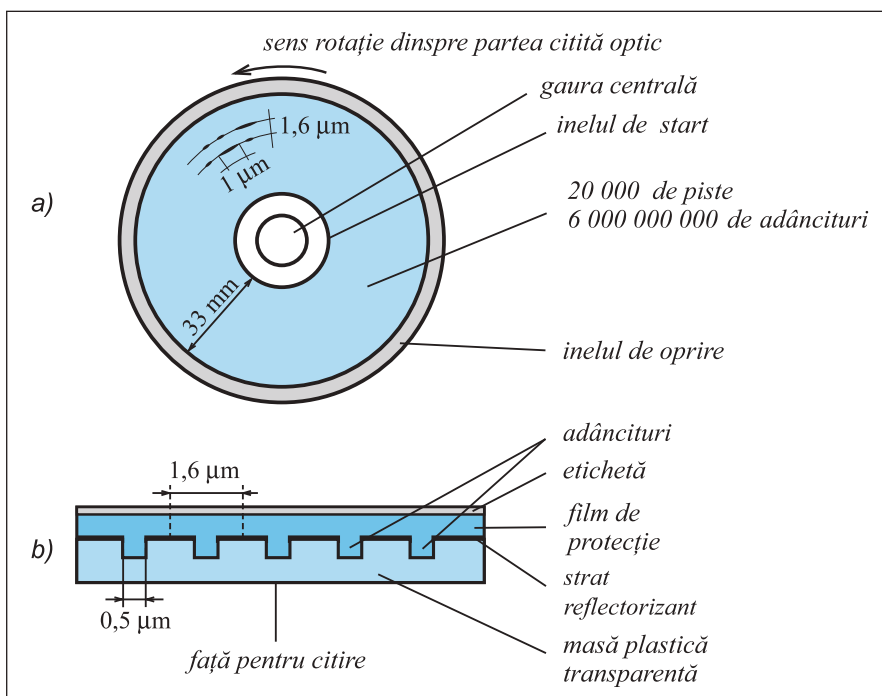


Fig. 6.9. Structura discului optic CD-ROM:
a – poziționarea pistelor pe disc;
b – secțiune a discului perpendiculară pe piste

Citirea discului optic se realizează cu ajutorul unui fascicul de lumină care, după ce se reflectă de pe suprafața activă, este interceptat de o celulă fotosensibilă (fig. 6.10).

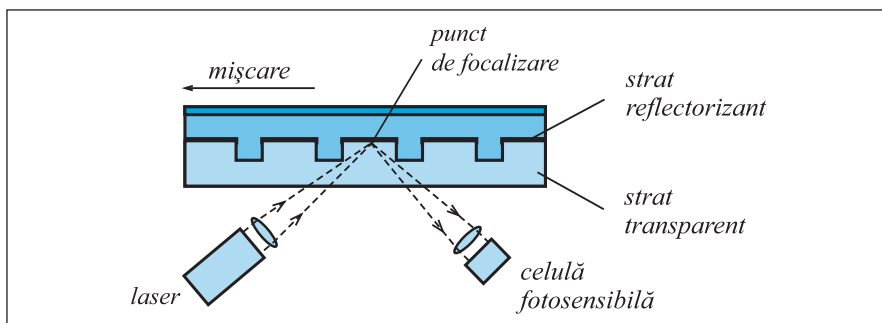


Fig. 6.10. Citirea discurilor optice

Parcurgând pisteles respective, fasciculul laser este reflectat când stratul reflectorizant se află în **punctul de focalizare** și nerefectat în caz contrar. Cu alte cuvinte, adânciturile de pe suprafața activă a discului optic schimbă (modulează) intensitatea fasciculului reflectat. În consecință, la ieșirea celulei fotosensibile se formează un semnal care redă succesiunea de cifre binare 0, 1 înregistrată pe disc la etapa fabricării.

În unitățile de disc optic moderne sursa de lumină – laserul și celula fotosensibilă – fotodiada se realizează într-un dispozitiv integral, denumit **cap optic de citire**. Viteza unghiulară a discului este de 200–600 de rotații pe minut, iar viteza liniară este de 1,4 m/s. În pofida vitezelor foarte mari, discul optic practic nu se uzează, întrucât între capul optic și disc nu există un contact mecanic direct. Prin urmare, durata de exploatare a unui disc *CD-ROM* este determinată de calitatea stratului reflectorizant și a stratului de protecție. În cazul utilizării alumiului, din cauza oxidării, stratul reflectorizant se întunecă, reducând astfel viața unui disc optic la 10-15 ani. În cazul unui strat reflectorizant din aur, numai o acțiune violentă care ar distruge stratul protector ar putea afecta calitatea unui disc optic.

De regulă, discurile optice *CD-ROM* se utilizează pentru tirajarea sistemelor de operare, a mediilor de programare, a enciclopediilor, jocurilor electronice și a altor informații destinate unui număr foarte mare de utilizatori.

Structura **discurilor optice inscriptibile și reinscriptibile** este redată în figura 6.11.

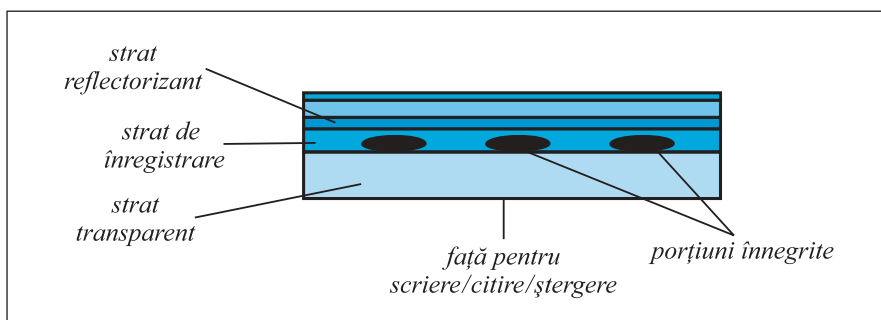


Fig. 6.11. Structura discurilor optice inscriptibile și reinscriptibile

Din figură se observă că discurile examinate conțin un strat special, denumit **strat de înregistrare**. În cazul **discurilor inscriptibile**, acest strat este format dintr-un material organic – *cyanin* sau *phtalocyanin*, care se întunecă la încălzire. Înregistrarea informației se realizează cu ajutorul unui fascicul laser foarte puternic care înnește porțiunile respective ale stratului de înregistrare.

La citire, porțiunile întinse blochează trecerea fasciculului laser spre stratul reflectorizant, modificând astfel intensitatea luminii captate de celula fotosensibilă (fig. 6.10). Întrucât fasciculul laser folosit la citire este foarte slab, discul inscriptibil poate fi citit de mai multe ori fără a distruge informația înregistrată.

În cazul **discurilor reinscriptibile**, materialul din care este format stratul de înregistrare devine din nou transparent dacă este încălzit până la o temperatură specială, denumită **temperatură critică**. Evident, după ștergere, pe discul reinscriptibil pot fi înscrise date binare noi. Discurile moderne *CD-RW* suportă până la 10 000 de cicluri de scriere/ștergere.

Neajunsul comun al discurilor inscriptibile și reinscriptibile este faptul că ele sunt mult mai sensibile la temperatură, durata lor de exploatare fiind mai mică decât a discurilor *CD-ROM*. Aceste discuri sunt și mai scumpe, întrucât pentru o citire sigură stratul reflectorizant este realizat din aur sau argint.

De obicei, discurile optice *CD-R* și *CD-RW* se utilizează pentru difuzarea informațiilor operative unui cerc bine determinat de utilizatori – baze de date, pachete specializate de programe, rapoartele unor congrese importante, tablourile unei expoziții, documente multimedia etc.

Menționăm faptul că realizările tehnologice din ultimul deceniu au condus la apariția discurilor optice de tipul *DVD* (*Digital Versatile Disc* – Disc Digital Multifuncțional). Capacitatea unui disc *DVD* este de aproape șapte ori mai mare decât a unui disc de tipul *CD*. Ulterior, au fost dezvoltate discuri și mai noi, de tipul *HVD* (*Holographic Versatile Disc* – Disc Holografic Multifuncțional). Capacitatea unui astfel de disc este de mii de ori mai mare decât a unui disc de tipul *CD*.

În prezent, în calculatoarele personale de uz general, discurile optice și, uneori, chiar și cele rigide, sunt înlocuite cu memorii de tip *flash*. În aceste memorii informația este stocată cu ajutorul unor circuite electronice, ce asigură scrierea, citirea și ștergerea datelor la viteze cu mult mai mari.

Întrebări și exerciții

- ① Cum sunt reprezentate cifrele binare 0, 1 în înregistrările optice?
- ② Care este capacitatea de memorare a unui disc optic?
- ③ Cum se înregistrează informația pe un disc optic *CD-ROM*?
- ④ De ce depinde capacitatea de memorare a unui disc optic?
- ⑤ E cunoscut faptul că lungimea spiralei pe care se înregistrează informația unui disc optic este de 5300 m. Capacitatea de memorare a discului este de 640 *Megaocteți*. Determinați densitatea de înregistrare a informației pe discul optic.
- ⑥ Care este domeniul de utilizare a discurilor *CD-ROM*?
- ⑦ De ce depinde timpul de acces la informațiile de pe un disc optic?
- ⑧ Viteza liniară a discului optic este de 1,4 m/s. Cunoscând densitatea de înregistrare a informației $d = 128$ *Kocteți/m*, determinați **viteza de transmisie a datelor** de la unitatea de disc la unitatea centrală. Amintim că viteza de transmisie a datelor se măsoară în *biți*, *Kbiți* sau *Mbiți* pe secundă.
- ⑨ Pe cele 20 000 de piste (spire) ale unui disc optic sunt înregistrați circa 640 de *Megaocteți*. Câtă informație conține o singură pistă a discului optic?
- ⑩ Pe un disc *CD-ROM* sunt înregistrate în formă binară circa 74 de minute de muzică. Determinați câtă informație va conține un cântec cu durata de 4 minute și 30 de secunde. Câte piste va ocupa cântecul respectiv?
- ⑪ Cum se citește informația de pe un disc optic? Care este destinația capului optic de citire?
- ⑫ Explicați destinația straturilor unui disc optic inscriptibil. Cum se realizează înregistrarea informației pe discul în cauză?
- ⑬ De ce depinde durata de exploatare a unui disc *CD-ROM*? Dar a discurilor *CD-R* și *CD-RW*?
- ⑭ Cum se înregistrează și se șterge informația de pe un disc optic reinscriptibil?
- ⑮ Care este domeniul de utilizare a discurilor *CD-R* și *CD-RW*?
- ⑯ **EXPERIMENTEAZĂ!** Aflați parametrii tehnici ai unităților de discuri optice

cu care, posibil, erau/sunt dotate calculatoarele din laboratorul de informatică al liceului dvs.

- ⑦ CREEAZĂ! Utilizând sursele de informare din Internet, scrieți un mic eseu despre evoluția discurilor optice.

6.8. Vizualizatorul și tastatura

Vizualizatorul este un dispozitiv de ieșire prin intermediul căruia informația este prezentată utilizatorului pe ecranul unui tub catodic. Schema funcțională a vizualizatorului este prezentată în figura 6.12.

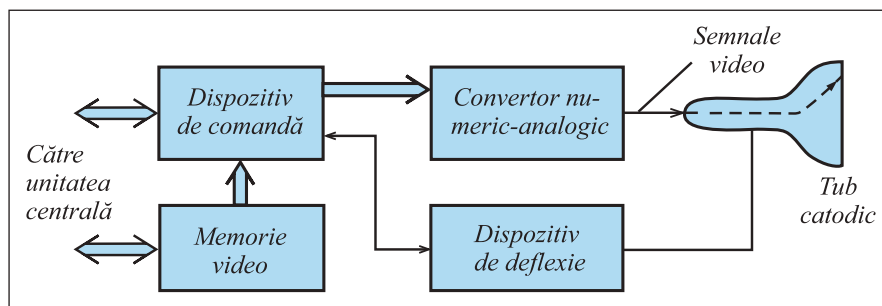


Fig. 6.12. Schema funcțională a vizualizatorului cu tub catodic

Ca și în cazul televizoarelor pentru publicul larg, imaginea pe ecranul tubului catodic se formează din puncte. Punctele sunt activate de un fascicul de electroni. Culoarea și luminanța fiecărui punct este controlată de semnale video, aplicate la intrările corespunzătoare ale tubului catodic. Parcurgerea punctelor într-o ordine prestabilită, de regulă, pe linii de la stânga la dreapta și de sus în jos, este asigurată de dispozitivul de deflexie.

Fiecărui punct al ecranului îi corespunde o locație în memoria video a vizualizatorului. Locațiile conțin informații referitoare la culoarea și luminanța punctelor respective. Dispozitivul de comandă citește locațiile memoriei video în ordinea parcurgerii punctelor de pe ecran. Conținutul fiecărei locații este interpretat drept o instrucțiune referitoare la modul de activare a punctului care îi corespunde. Semnalele video, necesare pentru comanda fascicului de electroni, sunt formate de convertoare numeric-analogice.

Memoria video a vizualizatorului este încărcată de unitatea centrală a calculatorului. În orice moment, calculatorul poate modifica conținutul acestei memorii. În consecință, se va schimba și imaginea afișată pe ecran. Imaginile pot fi animate prin modificarea conținutului memoriei video la frecvențe utilizate în cinematografie.

Vizualizatorul poate funcționa în unul dintre cele două regimuri: alfanumeric sau grafic.

În regim alfanumeric ecranul este împărțit în zone convenționale, numite **zone-caracter**. De regulă, aceste zone formează 25 de linii cu 80 de caractere pe linie. În fiecare zonă poate fi afișat un singur caracter dintr-un set de 256 de

caractere. Setul de caractere este constituit din literele mari și mici ale alfabetului latin, cifrele zecimale, simbolurile matematice, semnele de punctuație și unele caractere semigrafice, utilizate la afișarea pe ecran a tabelelor, diagramelor, chenarelor etc. Fiecare zonă-caracter poate avea o culoare pentru caracter și altă culoare pentru fundal, ceea ce îi permite utilizatorului să afișeze pe ecran texte cu litere de diferite culori.

În regim grafic utilizatorul poate controla afișarea pe ecran a fiecărui punct. Numărul de puncte pe orizontală și verticală determină rezoluția vizualizatorului. De exemplu, expresia „rezoluția 640×200” înseamnă că vizualizatorul are 640 de puncte pe orizontală și 200 pe verticală.

Există mai multe norme internaționale care reglementează rezoluția și numărul de culori ale vizualizatoarelor. Aceste caracteristici sunt recunoscute și respectate de firmele producătoare.

De exemplu, în cazul calculatoarelor personale, cele mai răspândite sunt normele denumite *EGA*, *VGA* și *SVGA*. Norma *EGA* stipulează că vizualizatorul are o rezoluție de 640×350 de puncte, permițând reprezentarea a 64 de culori.

Norma *VGA*, păstrând compatibilitatea cu *EGA*, oferă ca facilități suplimentare rezoluția 640×480 de puncte și 256 de culori distincte.

În scopul îmbunătățirii calității imaginii redată, norma *SVGA* adaugă rezoluția suplimentară de 1 024×768 de puncte.

În prezent, datorită realizărilor tehnologice din ultimii ani, ecranele cu tuburi catodice au fost înlocuite cu ecrane plate. Ecranul plat este alcătuit dintr-o matrice (un tablou) de celule emițătoare de lumină, câte o celulă pentru fiecare din micro-zonele imaginii digitale furnizate de calculator. Utilizarea celulelor emițătoare de lumină asigură o reducere substanțială a dimensiunilor vizualizatorului și o îmbunătățire semnificativă a rezoluției imaginilor afișate. Astfel de vizualizatoare se caracterizează printr-un consum redus de energie și rezoluții de până la $2\,560 \times 2\,048$ de puncte.

Tastatura este un dispozitiv de intrare care transformă acționarea unei taste într-un cuvânt binar, accesibil echipamentelor calculatorului.

Partea electronică a unei tastaturi constă dintr-un codificator. La intrările codificatorului se aplică semnalele logice provenite de la taste. La ieșire se furnizează cuvintele unui cod binar, de obicei standard (*ISO*, *ASCII*, *UNICODE* etc.). Unele tipuri de tastaturi pot fi prevăzute cu un generator audio, care la acționarea tastelor produce un sunet specific.

La noi, ca și în țările anglofone, cea mai răspândită este tastatura de tipul *QWERTY*, denumirea căreia provine de la amplasarea caracterelor *Q*, *W*, *E*, *R*, *T* și *Y* în rândul de sus al tastelor alfanumerice. În țările francofone se utilizează tastatura *AZERTY*, în Germania – *QWERTZ* etc.

Deși locul amplasării tastelor și numărul lor pot să difere, destinația tastelor este aceeași.

Tastatura dispune de următoarele grupe de taste: alfanumerice, funcționale și speciale. Grupul de taste alfanumerice include tastele cifrelor zecimale, tastele caracterelor alfabetului englez, tastele simbolurilor matematice și ale semnelor de punctuație. Grupul de taste funcționale include tastele <F1>, <F2>, ... , <F12>.

Tastele respective nu au o destinație prestabilită și semnificația lor este definită de programul care derulează pe calculator. Tastele speciale se utilizează pentru poziționarea cursorului vizualizatorului, pentru introducerea în calculator a unor cuvinte binare care nu au taste proprii etc.

În calculatoarele de performanță, vizualizatorul și tastatura pot forma un echipament periferic unitar, denumit **consolă**. Consola utilizată de operator pentru dirijarea proceselor de calcul se numește **monitor**.

În cazul calculatoarelor de tip tabletă și a celor încorporate, de exemplu, în telefoanele inteligente, echipamentele industriale cu dirijare numerică, electrocasnicele etc., în calitate de dispozitive de intrare-ieșire sunt utilizate **ecranele tactile** (*touchscreen*), care permit atât afișarea informațiilor vizuale, cât și introducerea comenzilor și datelor prin glisări și atingeri a elementelor de control afișate pe ecran.

Întrebări și exerciții

- 1 Explicați cum funcționează vizualizatorul. Care este destinația părților componente ale unui vizualizator?
- 2 Cum poate fi schimbată imaginea afișată pe ecranul unui vizualizator? Cum pot fi animate imaginile de pe ecran?
- 3 Prin ce se deosebesc regimurile de funcționare a vizualizatorului?
- 4 Care sunt indicii principali de calitate ai unui vizualizator?
- 5 EXPERIMENTEAZĂ! Determinați tipul vizualizatorului la care lucrați dvs. Aflați rezoluția și numărul de culori disponibile.
- 6 Care sunt părțile componente ale unei tastaturi? Cum se determină tipul tastaturii?
- 7 Numiți grupurile de taste și destinația lor.
- 8 CERCETEAZĂ! Utilizând sursele de informație din Internet, elaborați un studiu comparativ al vizualizatoarelor cu tub catodic și al celor cu ecran plat, punând în evidență atât avantajele, cât și neajunsurile fiecăruia din ele. Indicați domeniile de utilizare a fiecărui tip de vizualizator.
- 9 STUDIU DE CAZ. Cu ajutorul unui motor de căutare, găsiți pe Internet descrierile monitoarelor propuse de unitățile de comerț. Aflați cum parametrii tehnici ai monitoarelor influențează costurile, care sunt avantajele, neajunsurile și domeniile de utilizare a acestora.
- 10 CREEAZĂ! Utilizând Internetul în calitate de sursă de informație, scrieți un mic eseu despre evoluția vizualizatoarelor, atât a celor industriale, cât și a celor de uz personal.

6.9. Imprimantele

Imprimantele sunt dispozitive de ieșire care furnizează rezultatele sub forma unui document tipărit. În funcție de **tehnica de tipărire** utilizată, imprimantele se pot clasifica în:

– imprimante mecanice, în care tipărirea se face prin acționarea unor ciocănașe sau ace;

- imprimante laser, în care tipărirea se face folosind metode electrostatice ca la mașinile de copiat;
- imprimante cu jet de cerneală;
- imprimante termice, funcționarea cărora se bazează pe utilizarea unei hârtii speciale care își schimbă culoarea în funcție de temperatură.

Principiul de funcționare a **imprimantei matriciale cu ace** este prezentat în *figura 6.13*.

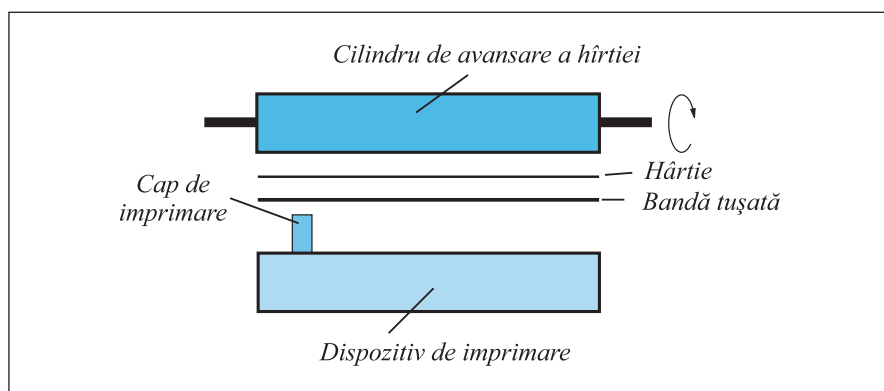


Fig. 6.13. Principiul de funcționare a imprimantei matriciale cu ace

Capul de imprimare conține un grup de ace metalice subțiri care la momentul potrivit lovesc prin bandă tușată în foaia de hârtie. Configurația ácelor ce lovesc în fiecare moment și avansarea capului de-a lungul liniei de imprimat determină imaginile tipărite.

Imprimantele matriciale pot funcționa în regim grafic și regim alfanumeric.

În **regim grafic**, calculatorul comandă tipărirea pe hârtie a fiecărui punct aparte. Evident, din puncte pot fi formate orice imagini: grafice, desene tehnice, caractere cu configurații concepute de utilizator. Imprimarea informației în regim grafic este lentă din cauza deplasărilor multiple ale capului de imprimare și a caracterului discontinuu al avansului de hârtie cu pași foarte mici.

În **regim alfanumeric**, calculatorul transmite imprimantei numai codurile caracterelor de tipărit. Fiecărui cod îi corespunde o imagine din puncte care se păstrează într-o memorie specială a imprimantei. Imaginile respective sunt tipărite prin una sau, cel mult, două-trei treceri ale capului de imprimare.

Imprimarea în regim alfanumeric este mult mai rapidă, însă pot fi tipărite numai caracterele imaginile cărora au fost, în prealabil, înmagazinate în memoria imprimantei. Imprimantele matriciale simple au câteva seturi de caractere, înscrise într-o memorie permanentă. Imprimantele matriciale mai performante permit încărcarea prin program a mai multe seturi de caractere cu configurații concepute de utilizator.

E cazul să subliniem că cu cât numărul ácelor din capul de imprimare este mai mare, cu atât calitatea imprimării este mai bună. În prezent se utilizează imprimante cu 9 sau cu 24 de ace. Calitatea imprimării poate fi îmbunătățită prin tipărirea repetată (de 2-4 ori) a aceluiași caracter pe același loc. Viteza de tipărire a imprimantelor mecanice cu ace este de 150-500 de caractere pe minut.

Principiul de funcționare a **imprimantelor laser** este prezentat în figura 6.14.

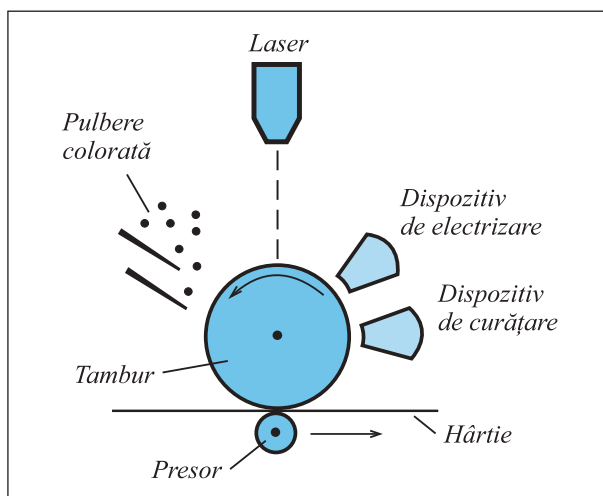


Fig. 6.14. Principiul de funcționare a imprimantei laser

Elementul principal al acestei imprimante este un tambur acoperit cu un strat semiconductor, care își poate schimba proprietățile electrice sub acțiunea luminii. Mai întâi, suprafața tamburului se electrizează. Cu ajutorul razelor laser pe suprafața electrizată a tamburului se proiectează punctele imaginii de tipărit. Întrucât sectoarele care au fost luminate își schimbă conductibilitatea, sarcinile electrice respective se neutralizează. Prin urmare, pe suprafața tamburului se formează o imagine electrică ascunsă. Developarea imaginii se efectuează cu ajutorul unor particule de pulbere colorată, atrase de sectoarele electrizate ale tamburului. Imaginile de pe tambur se transferă pe hârtie și se fixează prin topirea particulelor de pulbere.

În continuare, toate sarcinile electrice de pe suprafața tamburului sunt neutralizate, iar resturile de pulbere sunt curățate.

Imprimantele laser sunt cele mai bune dintre imprimantele moderne. Textele și grafica tipărite la aceste imprimante nu se deosebesc de cele tipografice. Viteza de tipărire este de 5-15 pagini pe minut.

Imprimantele cu jet de cerneală formează punctele unei imagini din picături microscopice pulverizate pe hârtie de niște ajutaje speciale. Ele asigură o calitate foarte bună a tiparului și sunt utilizate pentru imprimarea color. Aceste imprimante sunt mai scumpe decât imprimantele cu ace și necesită o îngrijire tehnică deosebită.

Imprimantele termice utilizează o matrice de ace a căror încălzire selectivă este determinată în funcție de caracterul ce urmează a fi tipărit. Viteza de imprimare este relativ mică, circa 300 de rânduri pe minut, însă avantajul principal constă în dimensiunile reduse ale imprimantei.

Întrebări și exerciții

- ❶ Cum se clasifică imprimantele în funcție de tehnica de tipărire?
- ❷ Care sunt părțile componente ale unei imprimante?

- ③ Care este principiul de funcționare a imprimantei matriciale cu ace? Cum funcționează această imprimantă?
- ④ Explicați cum funcționează o imprimantă laser. Care este avantajul principal al acestei imprimante?
- ⑤ **EXPERIMENTEAZĂ!** Determinați tipul imprimantei cu care lucrați dvs. Aflați parametrii tehnici ai imprimantei: setul de caractere, regimurile de funcționare, capacitatea memoriei-tampon, viteza de imprimare.
- ⑥ **STUDIU DE CAZ.** Cu ajutorul unui motor de căutare, găsiți pe Internet descrierile imprimantelor propuse de unitățile de comerț. Aflați cum parametrii tehnici ai imprimantelor influențează costurile, care sunt avantajele, neajunsurile și domeniile de utilizare a acestora.
- ⑦ **CREEAZĂ!** Utilizând Internetul în calitate de sursă de informație, scrieți un mic eseu despre evoluția imprimantelor, atât a celor industriale, cât și a celor de uz personal.

6.10. Clasificarea calculatoarelor

Caracteristica generală a unui calculator include următoarele date:

- viteza de operare;
- capacitatea memoriei interne;
- componența, capacitatea și timpul de acces ale unităților de memorie externă;
- componența și parametrii tehnici respectivi ai echipamentelor periferice;
- parametrii de masă și gabarit;
- costul.

În funcție de aceste date, calculatoarele moderne se clasifică în 4 categorii:

- supercalculatoare;
- calculatoare mari (macrocalculatoare);
- minicalculatoare;
- microcalculatoare (calculatoare personale).

Supercalculatoarele pot executa peste 10^{15} (1000 bilioane) de operații pe secundă, iar prețul lor depășește 20 milioane de dolari. Cercetări și proiectări în industria supercalculatoarelor se realizează în SUA și Japonia de firmele *IBM*, *Gray Research*, *Fujitsu*, *ETA Systems*, *Dell*, *Sutherland* etc. Supercalculatoarele se utilizează în prelucrări extrem de complexe ale datelor în aeronautică, fizica nucleară, astronomică, seismologie, prognoza meteo etc.

Calculatoarele mari (în engleză *mainframe* „dulapul principal”) pot executa sute de bilioane de operații pe secundă ($1 \text{ bilion} = 10^{12}$), prețul variind între 20 de mii și câteva milioane de dolari. Schema-bloc a calculatoarelor date este prezentată în *figura 6.2*. De regulă, calculatoarele mari includ zeci de unități de disc magnetic și imprimante, sute de console aflate la diferite distanțe de unitatea centrală. Aceste calculatoare se utilizează în cadrul unor mari centre de calcul și funcționează în regim non-stop. Principalele firme producătoare de calculatoare mari sunt *IBM*, *Hitachi*, *Amdahl*, *Fujitsu* etc.

Minicalculatoarele efectuau sute de milioane de operații pe secundă, iar prețul lor nu depășea 200-300 mii de dolari. Echipamentele periferice ale unui minicalculator includeau câteva discuri magnetice, una sau două imprimante, mai multe console. Minicalculatoarele erau mai ușor de utilizat și operat decât calculatoarele mari și se utilizau în proiectarea asistată de calculator, în automatizări industriale, pentru prelucrarea datelor în experimentele științifice etc. Dintre firmele ce produceau minicalculatoare vom remarca *IBM, Wang, Texas Instruments, Data General, DEC, Hewlett-Packard* etc. În prezent minicalculatoarele au fost înlocuite de calculatoarele personale.

Microcalculatoarele, denumite și calculatoare personale, sunt realizate la prețuri scăzute – între 100 și 15 000 de dolari și asigură o viteză de calcul de ordinul miliardelor de operații pe secundă. Schema-bloc a unui calculator personal este prezentată în figura 6.2.

De obicei, echipamentele periferice ale unui microcalculator includ una sau mai multe unități de disc rigid sau SSD (*Solid-State Drive*), o unitate de disc optic, o imprimantă și o consolă. Structura modulară și gruparea tuturor echipamentelor în jurul unei magistrale permite configurarea microcalculatorului în funcție de necesitățile individuale ale fiecărui utilizator.

În acest scop, microcalculatoarele dispun de câteva porturi USB (*Universal Serial Bus* – Magistrală Serială Universală), care permit conectarea diferitor echipamente periferice suplimentare, inclusiv a memoriilor de tip *flash*. De asemenea, calculatoarele personale, destinate plasării pe masa de lucru (de tip *desktop*), calculatoarele portabile (denumite *notebook*-uri sau *laptop*-uri) și cele de tip tabletă, sunt dotate cu dispozitive periferice ce asigură comunicarea cu alte sisteme tehnice digitale prin tehnologii radio (*Wi-Fi, Bluetooth*) sau raze infraroșii.

Corporații care produc microcalculatoare există în foarte multe țări, însă lideri mondiali, unanim recunoscuți, sunt firmele *Lenovo, Apple, Hewlett-Packard, Dell, Asus, Acer, Samsung* etc.

Întrebări și exerciții

- 1 EXPERIMENTEAZĂ! Numiți parametrii mai semnificativi ai unui calculator. Determinați parametrii respectivi ai calculatorului cu care lucrați dvs.
- 2 Cum se clasifică calculatoarele în funcție de parametrii tehnici și economici?
- 3 CREEAZĂ! În baza informațiilor colectate din Internet, dați o caracteristică succintă a fiecărei categorii de calculatoare: supercalculatoare, calculatoare mari, minicalculatoare și microcalculatoare. Puneți în evidență domeniile de utilizare a calculatoarelor din fiecare categorie.
- 4 EXPLOREAZĂ! Folosind un server de căutare, găsiți pe Internet informații detaliate despre principalii producători de calculatoare personale. Alcătuiți un clasament al acestor producători după cota de piață deținută de ei.
- 5 ANALIZEAZĂ! Identificați cota procentuală a profitului marilor companii ce produc calculatoare, pe categorii (supercalculatoare, calculatoare mari, microcalculatoare, minicalculatoare, microcalculatoare), și elaborați un mic studiu ce ar pune în evidență evoluția acestui sector important al economiei globale.

- ⑥ **CERCETEAZĂ!** Utilizând datele publicate de Biroul Național de Statistică și Agenția Națională pentru Reglementare în Comunicații Electronice și Tehnologie Informației (ANRCETI), elaborați un mic studiu despre producerea și comercializarea calculatoarelor în țara noastră.
- ⑦ **EXPLOREAZĂ!** Un grup internațional de specialiști în domeniul informaticii implementează așa-numitul proiect „Top 500”. Fiind lansat în 1993, acest proiect publică de două ori pe an lista primelor 500 celor mai puternice supercalculatoare. Elaborati un mic studiu despre primele zece supercalculatoare, punând în evidență parametrii tehnici și destinația acestora, companiile care le produc și rolul cooperării internaționale în dezvoltarea științei calculatoarelor și tehnologiilor informației.

6.11. Microprocesorul

Microprocesorul este un circuit integrat care implementează funcțiile unității centrale de prelucrare a informației, și anume – extragerea și executarea instrucțiunilor.

De regulă, un microprocesor conține un dispozitiv aritmetic și altul de comandă, un grup de registre destinate păstrării temporale a datelor frecvent utilizate, magistrale și circuite de comandă aferente (fig. 6.15).

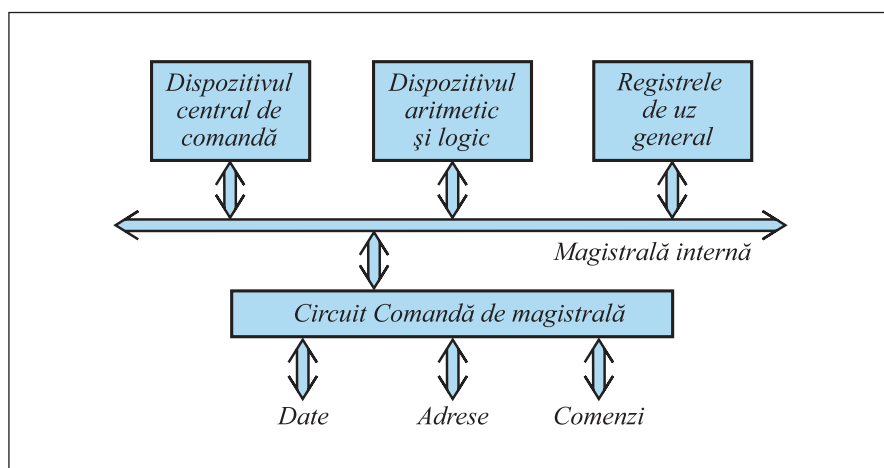


Fig. 6.15. Schema funcțională a unui microprocesor

Microprocesorul interacționează cu unitățile de memorie și echipamentele periferice prin intermediul a trei magistrale: *Date*, *Adrese* și *Comenzi*. Traficul de informații prin magistrale este controlat de circuitul *Comandă de magistrală*.

Extragerea și executarea instrucțiunilor are loc sub controlul **dispozitivului central de comandă**. Pentru aceasta, prin magistrala de adrese se indică adresa locației memoriei interne, iar prin magistrala de comenzi se transmit semnalele necesare de scriere sau citire. Datele citite sau scrise se transmit prin magistrala de date.

Microprocesoarele se apreciază după următorii parametri:

- lungimea cuvântului;
- frecvența ceasului de sistem;
- capacitatea magistralelor.

Lungimea cuvântului reprezintă numărul de biți ai succesiunilor binare care pot fi memorizate în registre și prelucrate de dispozitivul aritmetic al microprocesorului. Microprocesoarele moderne au lungimea cuvântului de 32, 64 și mai mulți biți.

Frecvența ceasului de sistem reprezintă numărul de impulsuri pe secundă pe care le produce generatorul de impulsuri de tact din componența dispozitivului de comandă. Frecvența ceasului de sistem se măsoară în **Megahertz** ($1\text{ MHz} = 10^6\text{ Hz}$). Întrucât impulsurile de tact sincronizează executarea microoperațiilor în toate dispozitivele microprocesorului, frecvența ceasului de sistem caracterizează rapiditatea microprocesorului.

Capacitatea magistralei reprezintă numărul de biți ai succesiunilor binare transmise prin magistrală. De obicei, capacitatea magistralei de date este egală sau mai mare decât lungimea cuvântului microprocesorului. În caz contrar, pentru a transmite un cuvânt, sunt necesare mai multe cicluri ale magistralei de date.

Capacitatea magistralei de adrese determină spațiul de adrese care pot fi accesate direct de microprocesor. Astfel, un microprocesor, având capacitatea magistralei de adrese de 16 biți, poate accesa 2^{16} locații ale memoriei interne, iar un microprocesor cu capacitatea magistralei de adrese de 32 de biți poate accesa direct 2^{32} locații.

În ansamblu, lungimea cuvântului, frecvența ceasului de sistem și capacitatea magistralelor determină **capacitatea de prelucrare a microprocesorului**, care se măsoară în **Mips** – Megainstrucțiuni pe secundă. Pentru exemplificare, în *tabelul 6.2* sunt prezentate caracteristicile principale ale microprocesoarelor din familia *Intel*.

Tabelul 6.2

**Caracteristicile principale
ale microprocesoarelor din familia *Intel***

Microprocesor	Lungime cuvânt, <i>bit</i>	Frecvență, <i>MHz</i>	Capacitate de prelucrare, <i>Mips</i>
4040 (anul 1974)	4	0,7	0,06
8085 (anul 1976)	8	5	1,25
80286 (anul 1982)	16	16	8
Pentium 4 (anul 2004)	32	3800	2500
Pentium D (anul 2005)	64	3400	4000
Intel Core i9 (anul 2017)	64	4500	5500

Din tabelul de mai sus se observă că pe parcursul anilor 1974–2017 lungimea cuvintelor cu care operează un microprocesor a crescut de 8 ori, frecvența ceasului de sistem – de circa 6 500 de ori, iar capacitatea de prelucrare a acestora – de circa 9 200 de ori. Capacitatea de prelucrare a unui microprocesor modern este de circa 6 000 de ori mai mare decât a calculatorului *Modulului Lunar Apollo*, cu care în anul 1969 primii oameni de pe Pământ au aselenizat pe Lună.

Tipul microprocesorului și frecvența ceasului de sistem al calculatorului personal cu care lucrați dvs. poate fi aflat cu ajutorul meniului contextual al pictogramei „My computer”. Amintim că acest meniu se afișează pe ecran cu ajutorul unui clic-dreapta pe pictograma respectivă.

Întrebări și exerciții

- ❶ Explicați destinația componentelor unui microprocesor (*fig. 6.15*).
- ❷ Care sunt parametrii principali ai unui microprocesor? Explicați semnificația fiecărui parametru.
- ❸ EXPERIMENTEAZĂ! Determinați tipul și parametrii principali ai microprocesorului din componența calculatorului personal cu care lucrați dvs.
- ❹ EXPLOREAZĂ! În afară de Intel, în calculatoarele moderne se utilizează și microprocesoare produse de alte companii. Analizând informațiile despre calculatoarele personale comercializate în țara noastră, identificați producătorii de microprocesoare din componența acestora. Observați cum variază costul calculatoarelor personale în funcție de microprocesoarele utilizate.
- ❺ CREEAZĂ! Elaborați un mic studiu despre microprocesoarele din componența calculatoarelor de tip *desktop*, *notebook/laptop*, tabletă și a celor încorporate în telefoanele inteligente, frecvent utilizate în țara noastră.
- ❻ CERCETEAZĂ! Formulată în anul 1965, „legea” lui *Moore*, fondatorul companiei Intel, descrie o tendință pe termen lung în istoria calculatoarelor: numărul de tranzistori care pot fi plasați pe un circuit integrat se dublează aproximativ la fiecare doi ani. Această tendință a continuat de mai bine de o jumătate de secol. Analizând datele referitoare la numărul de tranzistori din componența circuitelor integrate, publicate pe Internet, aflați dacă această „lege” descrie corect evoluția microprocesoarelor din ultimii ani.

Capitolul 7

REȚELE DE CALCULATOARE

7.1. Introducere în rețele

Odată cu extinderea domeniilor de aplicare a calculatoarelor, a crescut și numărul utilizatorilor ce doreau să aibă acces la mijloace eficiente de prelucrare și stocare a unor informații comune.

De exemplu, în cazul proiectării unei clădiri, mai mulți specialiști – arhitectul, inginerul, pompierul ș.a., doresc să aibă acces și să introducă, dacă e necesar, modificări în desenele tehnice ale construcției în curs de elaborare. În cazul unei companii de transporturi aeriene, biletele la una și aceeași cursă pot fi vândute de mai multe agenții aflate în orașe diferite.

Soluția inițială la astfel de probleme a constituit-o conectarea la un calculator central, de mare capacitate, a mai multe terminale (fig. 7.1).

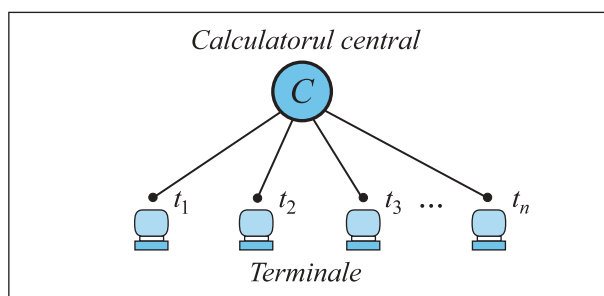


Fig. 7.1. Sistem centralizat de calcul

De regulă, un terminal constă dintr-un vizualizator, o tastatură și, dacă e necesar, o imprimantă. Neajunsul principal al unui sistem centralizat de prelucrare a datelor este fiabilitatea redusă și utilizarea inefficientă a resurselor de calcul.

Cu timpul, a apărut tendința de trecere de la sistemele centralizate la instalarea de calculatoare la fiecare utilizator și asigurarea unor legături de comunicație eficiente între ele (fig. 7.2).

Numim rețea de calculatoare o mulțime de calculatoare ce pot schimba informații prin intermediul unei structuri de comunicație.

Calculatoarele unei rețele se conectează la structura de comunicație prin intermediul unor unități de intrare-ieșire dedicate, numite **adaptoare de rețea**. Evident, în cadrul unei rețele fiecare calculator, mai exact, fiecare adaptor de rețea are o adresă unică, denumită **adresă de rețea**.

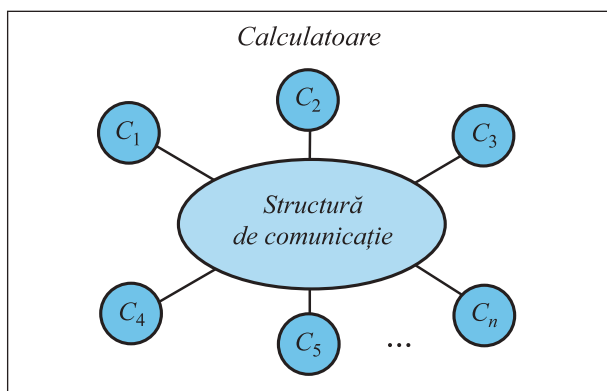


Fig. 7.2. Rețea de calculatoare

De exemplu, o rețea de calculatoare poate fi construită utilizând ca **structură de comunicație** rețeaua existentă de telefoane. În acest caz, adaptorul de rețea va include un modulator pentru conversiunea semnalelor digitale furnizate de calculator în semnale telefonice și un demodulator pentru operația inversă. Dispozitivul respectiv poartă denumirea de **modem** (**modulator-demodulator**). Adresa de rețea este dată de numărul de telefon al postului la care este conectat modemul.

În general, o structură de comunicație este formată din **linii de transmisie** a semnalelor. Aceste linii pot fi:

- cabluri cu fire torsadate;
- cabluri coaxiale;
- cabluri optice;
- linii cu microunde (terestre sau prin satelit).

Cablurile cu fire torsadate sunt asemănătoare celor telefonice și asigură o capacitate de transmisie de până la 1 *Mbit/s*. **Cablurile coaxiale**, asemănătoare celor din rețelele de televiziune prin cablu, asigură o capacitate de transmisie de până la 1 *Gbit/s*. **Cablul optic** constă din fibre de sticlă sau din plastic transparent, acoperite cu un înveliș de protecție. Semnalul optic, emis de o sursă laser, se propagă prin fibră și este recepționat de o celulă fotosensibilă. Capacitatea de transmisie a unui cablu optic poate ajunge la valoarea de 1 *Tbit/s*.

Liniile cu microunde sunt formate din stații de retransmisie ce operează în banda de unde centimetrice. Pe Pământ aceste stații se amplasează în raza vizibilității directe a antenelor, la o distanță de 40–50 de kilometri una de alta. În cazul liniilor cosmice, stațiile respective se amplasează pe sateliți. Capacitatea de transmisie a liniilor cu microunde este de ordinul 10 *Gbit/s*.

În funcție de **aria de răspândire** a calculatoarelor dintr-o rețea, există următoarele **tipuri de rețele**:

- rețele locale;
- rețele regionale;
- rețele globale.

În **rețelele locale** calculatoarele au o arie mică de răspândire (până la 2 km) și deservesc o singură instituție. Rețelele locale sunt formate, de regulă, din calculatoarele care se află în aceeași clădire sau într-un grup de clădiri. De obicei,

ca linii de transmisie se utilizează cablurile cu fire torsadate și cablurile coaxiale.

În ultimii ani, în cazul rețelelor locale mici și foarte mici, o răspândire tot mai largă o au mijloacele ce asigură comunicarea echipamentelor digitale prin tehnologii radio (*Wi-Fi*, *Bluetooth*) sau raze infraroșii.

Rețelele regionale acoperă aria unui oraș sau a unui sector. Liniile de comunicație se realizează prin cabluri coaxiale sau stații mici de transmisie/recepție, denumite **radiomodemuri**.

Rețelele globale acoperă suprafața unei țări, suprafața unui continent sau chiar suprafața mai multor continente. Ca linii de transmisie se utilizează cablurile optice și liniile cu microunde (terestre sau prin satelit).

Avantajul principal al rețelelor constă în **partajarea** sau, cu alte cuvinte, **utilizarea în comun** a datelor, a programelor și a calculatoarelor din rețea.

De exemplu, în cazul unei rețele locale, pot fi partajate fișierele, discurile de capacitate mare, imprimantele, cititoare de desene și alte periferice. Evident, fiind accesibile pentru mai mulți utilizatori, echipamentele periferice respective vor fi utilizate mai eficient. Totodată, specialiștii instituției în cauză pot lucra în echipă asupra unor proiecte comune: bugetul anual, planul de vânzări, actualizarea bazelor de date etc.

În cazul rețelelor globale, colective de cercetători din diferite țări pot efectua calcule complexe pe un supercalculator unic în lume sau analiza în comun rezultatele unui experiment științific foarte costisitor. Pe baza rețelelor examinate sunt create diverse servicii: transferul de fișiere, poșta electronică, difuzarea noutăților, conversații pe grupuri de interese, jocuri electronice, publicitate, transferul banilor etc.

Întrebări și exerciții

- ❶ Numiți factorii care au contribuit la apariția rețelelor de calculatoare.
- ❷ Care sunt neajunsurile sistemelor centralizate de calcul?
- ❸ Numiți componentele principale ale unei rețele de calculatoare.
- ❹ Explicați destinația structurii de comunicație.
- ❺ Care sunt funcțiile adaptorului de rețea? Cum se identifică calculatoarele din componența unei rețele? Determinați tipul adaptorului de rețea cu care lucrați dvs.
- ❻ Din ce este formată o structură de comunicație?
- ❼ Care este destinația unui modem? Dar a unui radiomodem?
- ❽ Numiți capacitățile de transmisie a următoarelor linii de comunicație:
 - cablu cu fire torsadate;
 - cablu coaxial;
 - cablu optic;
 - linie cu microunde.
- ❾ Estimați durata de transmisie a unui film video ($\approx 800 \text{ Gbiți}$) prin liniile de comunicație pe care le cunoașteți dvs.
- ❿ EXPERIMENTEAȚĂ! Determinați tipul liniilor de comunicație din structura rețelelor cu care lucrați dvs.
- ⓫ Cum se clasifică rețelele în funcție de aria de răspândire?

- 12 EXPERIMENTEAZĂ! Determinați tipul rețelei (locală, regională sau globală) cu care lucrați dvs.
- 13 Care sunt avantajele rețelelor de calculatoare? Ce servicii oferă o rețea de calculatoare?
- 14 EXPLOREAZĂ! Aflați capacitățile de transmisie a următoarelor mijloace de comunicație din componența rețelelor de calculatoare:
- *Wi-Fi*;
 - *Bluetooth*;
 - cu raze infraroșii.

7.2. Tehnologii de cooperare în rețea

Resursele unei rețele de calculatoare sunt echipamentele periferice, liniile de comunicație, calculatoarele propriu-zise, fișierele, bazele de date, programele executabile etc. Utilizarea eficientă a acestor resurse presupune lucrul în comun sau, cu alte cuvinte, cooperarea calculatoarelor și programelor ce rulează pe ele.

Numim tehnologie de cooperare modul cum este organizată funcționarea în comun a calculatoarelor și programelor din rețea.

Cel mai frecvent în rețelele de calculatoare se utilizează tehnologiile client-server și egal-la-egal.

În **tehnologia client-server** o resursă comună, de exemplu, imprimanta color sau discul de mare capacitate, este gestionată de un calculator dedicat, denumit **server**. Calculatorul care dorește să aibă acces la aceste resurse se numește **client**. Pentru a utiliza resursele respective, orice client trimite serverului o cerere. Serverul analizează cererile primite și, în funcție de statutul fiecărui client, le acceptă sau le respinge (fig. 7.3).

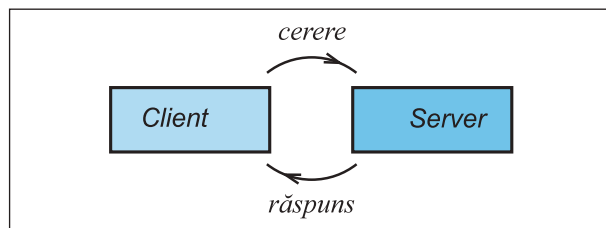


Fig. 7.3. Modelul client-server

Evident, calculatorul care gestionează fișierele comune se va numi **server de fișiere**, iar cel care gestionează imprimantele – **server de imprimare**.

Calculatorul care gestionează liniile de comunicație, alte resurse comune și, posibil, asigură accesul la rețele externe se numește **server de rețea**. De regulă, anume pe acest calculator rulează și sistemul de operare al rețelei. Celelalte calculatoare din rețea au caracteristici mai modeste și se numesc **stații de lucru**.

Avantajele principale ale tehnologiei client-server sunt:

- utilizarea eficientă a echipamentelor scumpe;

- distribuirea rațională a lucrărilor între calculatoare în funcție de puterea lor;
- protecția sporită a datelor importante, ele fiind păstrate numai pe servere.

Cu regret, tehnologiile client-server sunt complicate și necesită calculatoare performante.

În **tehnologia egal-la-egal** funcțiile tuturor calculatoarelor din rețea sunt identice. Fiecare calculator din rețea funcționează atât ca server, cât și ca stație de lucru, oferind pentru uzul public resursele de care dispune, în mod curent, unele fișiere de pe discul fix, unitatea de disc optic, imprimanta etc. Fiind simplă, tehnologia dată se folosește în rețelele locale mici. În cazul rețelelor mari, tehnologia egal-la-egal nu permite protecția sigură a datelor.

În mod similar, tehnologiile client-server se aplică și pentru organizarea lucrului în comun a două sau a mai multe programe.

Programul în curs de execuție care oferă servicii se numește program server, iar programele care apelează la aceste servicii se numesc programe client.

De exemplu, un program server ce gestionează o bază de date execută următoarele funcții:

- asigură protecția și securitatea datelor;
- recepționează și, dacă clientul are autorizațiile respective, execută cererile de modificare a datelor;
- recepționează cererile de citire a datelor și, în funcție de statutul clientului, permite sau interzice accesul la datele respective;
- completează un registru în care înscrie toate operațiile efectuate în baza de date.

Programul client asigură interacțiunea utilizatorului cu baza de date și realizează următoarele funcții:

- oferă utilizatorului o interfață simplă și comodă;
- verifică și editează datele introduse de utilizator;
- adresează cereri programului server;
- afișează informațiile primite din baza de date.

Programele server și programele client pot rula pe un singur calculator sau pe calculatoare diferite. În ultimul caz, prelucrarea de date este **distribuită**. Transferul de date între programul client și programul server se realizează prin structura de comunicație (fig. 7.2). Calculatorul pe care rulează un program server se numește **gazdă** (în engleză *host*).

De obicei, în rețelele locale programul client se execută pe stațiile de lucru, iar programele server rulează pe serverul de rețea. În cazul rețelelor regionale sau globale, pe fiecare calculator performant rulează mai multe programe server ce oferă diverse servicii programelor client de pe alte calculatoare.

Întrebări și exerciții

- 1 Explicați termenul *tehnologii de cooperare în rețea*. Ce tehnologii de cooperare în rețea cunoașteți?
- 2 Cum este organizată funcționarea calculatoarelor din rețea în cazul tehnologiei client-server?

- ③ Care sunt avantajele și dezavantajele tehnologiei client-server?
- ④ Explicați destinația serverelor de rețea, a serverului de fișiere, a serverului de imprimare și a stației de lucru.
- ⑤ Cum este organizată funcționarea calculatoarelor din rețea în cazul tehnologiei egal-la-egal? Care sunt avantajele și dezavantajele acestei tehnologii?
- ⑥ EXPERIMENTEAZĂ! Determinați tehnologiile de cooperare realizate în rețeaua cu care lucrați dvs. Există oare în rețea un server de fișiere și/sau un server de imprimare?
- ⑦ Există în rețeaua cu care lucrați dvs. un server de rețea? Argumentați răspunsul.
- ⑧ Este oare realizată în rețeaua cu care lucrați dvs. tehnologia egal-la-egal? Argumentați răspunsul.
- ⑨ Cum interacționează programele în curs de execuție în cazul tehnologiei client-server?
- ⑩ Numiți funcțiile unui program server și ale unui program client.
- ⑪ Explicați termenul *calculator-gazdă*.
- ⑫ CERCETEază! O companie de transporturi aeriene a deschis agenții de vânzare a biletelor în mai multe orașe din țară. Datele despre toate rutele aeriene și locurile disponibile se păstrează în calculatorul din sediul central al companiei. Care tehnologii de rețea ar asigura lucrul eficient al agențiilor de vânzare a biletelor? Sunt oare necesare programe server și programe client? Ce funcții ar realiza aceste programe?
- ⑬ ANALIZEAZĂ! Jocurile electronice colective presupun existența a 5-10 calculatoare reunite în rețea. În astfel de jocuri fiecare luptă contra tuturor. Calculatoarele respective oferă pentru uzul public o partiție de pe discul fix. Ce tehnologie de rețea ar asigura lucrul în comun al calculatoarelor?
- ⑭ PROIECTEAZĂ! Elaborați o tehnologie de cooperare în rețea pentru calculatoarele din:
 - a) depozitele unei firme;
 - b) sălile de expoziție ale unui muzeu;
 - c) sălile de lectură ale unei biblioteci;
 - d) casele unui magazin care acceptă carduri bancare;
 - e) sistemul de eliberare a banilor lichizi prin intermediul automatelor bancare;
 - f) sistemul de verificare operativă a numerelor de înmatriculare a automobilelor (fiecare echipaj de poliție este dotat cu un microcalculator și un radiomodem);
 - g) laboratorul de informatică.

7.3. Topologia și arhitectura rețelelor

Structura de comunicație a unei rețele (*fig. 7.2*) asigură transferul de date între calculatoare. De obicei, datele de transmis sunt grupate în pachete.

Un **pachet de date** conține următoarele informații:

- adresa destinatarului;
- datele propriu-zise;
- informații de control;
- adresa expeditorului.

Se observă că pachetul de date poate fi tratat ca un plic obișnuit care circulă într-un sistem tradițional de poștă. Drumul parcurs de un pachet de date depinde de topologia rețelei.

Numim topologie a rețelei configurația geometrică a legăturilor dintre calculatoare.

Topologiile concrete ale rețelelor actuale sunt formate prin utilizarea structurilor de bază: stea, inel, magistrală, distribuită etc. (fig. 7.4).

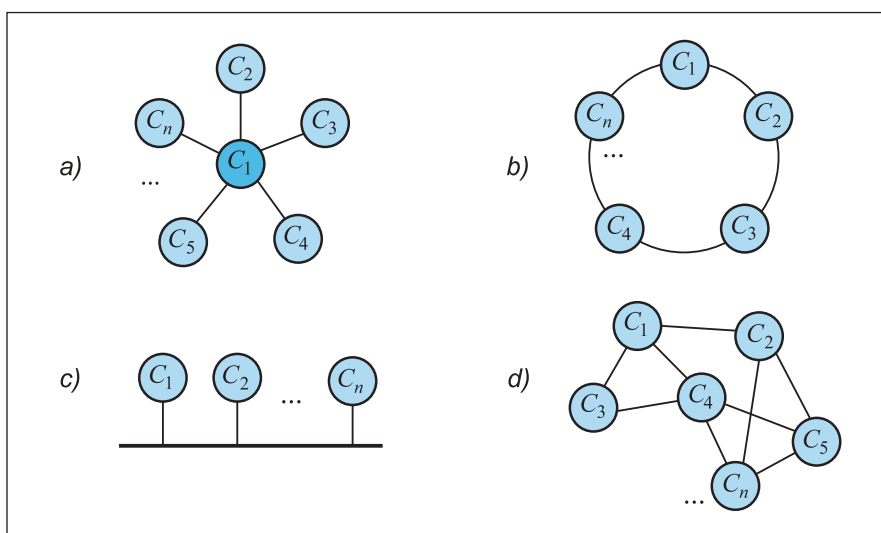


Fig. 7.4. Topologii de rețea:
a – stea; b – inel; c – magistrală; d – distribuită

În cazul **topologiei stea**, legătura dintre două calculatoare C_i , C_j ale rețelei are loc prin intermediul calculatorului central C_1 . Din acest motiv, calculatorul C_1 , denumit **calculator principal**, are un rol important în funcționarea rețelei, efectuând dispecerizarea pachetelor de date. Evident, defectarea calculatorului principal întrerupe funcționarea întregii rețele. Prin urmare, calculatorul principal trebuie să fie foarte fiabil.

În **topologia inel** legăturile dintre calculatoare formează o buclă închisă. Pachetul expedit de calculatorul C_i este transmis calculatorului C_{i+1} , care, la rândul său, îl transmite calculatorului C_{i+2} etc., până când pachetul ajunge la calculatorul destinat C_j . Întrucât defectarea oricărui calculator întrerupe funcționarea întregii rețele, toate calculatoarele C_1 , C_2 , ..., C_n trebuie să fie foarte fiabile.

În **topologia magistrală** există un singur canal de comunicație la care sunt conectate toate calculatoarele. Fiecare calculator „spionează” magistrala și interceptează pachetele adresate lui. Orice calculator C_i poate expedia un pachet numai atunci când magistrala este liberă.

Rețelele bazate pe topologia de tip magistrală sunt foarte fiabile, întrucât comunicarea dintre calculatoarele C_i , C_j va avea loc chiar și în cazul în care toate celelalte calculatoare nu funcționează.

În **topologiile distribuite** între fiecare pereche de calculatoare există mai multe căi de transmisie a datelor. De exemplu, un pachet de date expediat de calculatorul C_1 calculatorului C_n (fig. 7.4d) poate ajunge la destinație pe drumul $C_1 - C_2 - C_5 - C_n$, iar altul pe drumul $C_1 - C_4 - C_n$. Evident, rețeaua rămâne funcțională chiar dacă unul sau mai multe calculatoare și linii de comunicație nu funcționează.

De regulă, topologiile de tip stea, inel sau magistrală se utilizează în cazul rețelelor locale. Rețelele regionale și cele globale au o topologie distribuită. Integrarea rețelelor locale în rețele regionale și rețele globale se face conform topologiilor de bază din figura 7.4, fiecare nod $C_1, C_2, \dots, C_n$ reprezentând o subrețea. Prin urmare, ajungem la o structură ierarhică cu o diversitate de legături.

Setul de reguli pentru gestionarea schimbului de date într-o rețea este numit protocol de comunicație sau, pur și simplu, protocol.

Un **protocol** definește modul de adresare a calculatoarelor, lungimea și componența pachetelor de date, algoritmul de depistare și corectare a erorilor, modul de conectare fizică a adaptoarelor și cablurilor de rețea etc. La apariția primelor rețele de calculatoare fiecare producător de echipamente de calcul avea propriile sale protocoale de comunicație, ceea ce făcea imposibilă interconectarea calculatoarelor de proveniențe diferite. Acest neajuns a fost înlăturat prin standardizarea protocoalelor. Amintim că **standardul** reprezintă un document în care se reglementează calitatea, caracteristicile, forma etc. ale unui produs. Standardele internaționale sunt elaborate de Organizația Internațională pentru Standardizare (ISO – *International Standards Organisation*). Un alt organism internațional care joacă un rol important în standardizarea în domeniile electronicii și tehnicii de calcul este Institutul Inginerilor Electricieni și Electroniști (IEEE – *Institute of Electrical and Electronics Engineers*).

Numim arhitectură a rețelei modul în care este concepută rețeaua: topologia, protocoalele de comunicație, tehnologiile de cooperare în rețea.

În continuare prezentăm cele mai răspândite arhitecturi de rețele.

Ethernet (rețea în eter) – rețele locale realizate în conformitate cu standardul IEEE 802.3. Utilizează o magistrală din cablu cu fire torsadate, cablu coaxial sau cablu cu fibre optice. Viteza de transmisie a datelor ajunge la 100 Mbit/s. Arhitectura a fost elaborată de firmele XEROX, Intel și DEC.

Token-Ring (inel cu jeton) – rețele locale realizate în conformitate cu standardul IEEE 802.5. Utilizează un inel din cablu cu fire torsadate sau cablu coaxial. Viteza de transmisie a datelor este de 16 Mbit/s. Arhitectura a fost elaborată de firma IBM.

DATAKIT – rețele locale, regionale sau globale elaborate de firma Bell Laboratories. Sub aspect topologic, constă dintr-o mulțime de stele interconectate. În cazul cablului cu fibre optice, se atinge viteza de transmisie de 1,5 Gbit/s.

SNA (Sistem Network Architecture) – o arhitectură elaborată de firma IBM pentru rețelele locale, regionale și globale. Protocoalele acestei arhitecturi au stat la baza standardelor ISO. Topologia inițială era de tip stea, în prezent este o topologie distribuită, suportând și rețele locale.

ARPANET – o arhitectură concepută de mai multe universități și corporații sub egida Ministerului Apărării al SUA (*Advanced Research Projects Agency*). Arhitectura dată se bazează pe o topologie distribuită și folosește diferite linii de comunicație, de la liniile telefonice până la liniile cu microunde prin satelit. Liniile respective conectează supercalculatoare separate și diverse rețele locale sau regionale, răspândite pe aproape jumătate din suprafața terestră. Arhitectura **ARPANET** include următoarele protocoale:

- protocolul **IP** (*Internet Protocol*) destinat interconectării rețelelor locale, regionale și globale;
- protocolul serviciilor bazate pe conexiuni, numit **TCP** (*Transmission Control Protocol*);
- un protocol de transfer de fișiere, numit **FTP** (*File Transfer Protocol*);
- un protocol pentru poșta electronică, numit **SMTP** (*Simple Mail Transfer Protocol*);
- un protocol de atașare de la distanță a calculatoarelor, numit **TELNET**.

Pe baza arhitecturii **ARPANET** a fost concepută rețeaua globală de calculatoare *Internet*, care va fi studiată în paragraful următor.

Accentuăm faptul că odată cu încorporarea microcalculatoarelor în cele mai diverse mașini, aparate și dispozitive, în particular, în cele electrocasnice, a apărut posibilitatea conectării acestora în rețele, denumite *rețele de obiecte* sau *Internet al obiectelor* (în engleză *Internet of Things*, abreviat *IoT*).

Obiectele conectate la *Internetul obiectelor* pot fi controlate la distanță, fapt ce oferă noi posibilități de creștere a productivității muncii și îmbunătățirii vieții oamenilor. Totodată, răspândirea tot mai largă a acestor tehnologii este însoțită de apariția unor probleme importante, legate de securitatea rețelelor și obiectelor conectate la ele. Eventualele amenințări de securitate se referă nu doar la confidențialitatea și integritatea informațiilor procesate, dar și la eventualele daune materiale și umane, care pot fi provocate de obiectele ieșite de sub control sau de cele controlate de intruși.

Întrebări și exerciții

- ❶ Explicați termenul *pachet de date*. Ce informații conține un pachet de date?
- ❷ Explicați termenul *topologia rețelei*.
- ❸ **STUDIUL DE CAZ.** Care sunt topologiile de bază ale rețelelor? Numiți avantajele și dezavantajele fiecărei topologii de bază.
- ❹ Explicați cum se transmit pachetele de date în următoarele topologii de bază: stea, inel, magistrală, distribuită.
- ❺ Precizați drumul pe care îl va parcurge un pachet trimis de calculatorul C_2 calculatorului C_4 (fig. 7.4a).
- ❻ Găsiți drumul pe care îl parcurg pachetele expediate de calculatorul C_2 calculatorului C_4 (fig. 7.4b).
- ❼ Câte căi de transmitere a pachetelor există între calculatoarele C_1 , C_n din figura 7.4d? Care este calea cea mai scurtă?
- ❽ Ce se va întâmpla cu rețelele din figura 7.4 dacă iese din funcție calculatorul C_1 ? Dar dacă iese din funcție calculatorul C_2 ?

- 9 EXPERIMENTEAZĂ! Determinați topologia rețelei locale cu care lucrați dvs. Enumerați avantajele și dezavantajele acestei topologii.
- 10 Care este destinația unui protocol? Ce norme conține un protocol?
- 11 Argumentați necesitatea standardizării protocoalelor. Cine elaborează standardele respective?
- 12 EXPERIMENTEAZĂ! Aflați protocoalele utilizate de rețeaua locală cu care lucrați dvs.
- 13 Explicați modul cum se integrează rețelele locale în rețelele regionale și cele globale. Desenați topologia rețelelor regionale sau globale la care aveți acces dvs.
- 14 Explicați termenul *arhitectura rețelei*.
- 15 Dați exemple de arhitecturi ale rețelelor locale, regionale și globale.
- 16 Caracterizați arhitectura rețelei cu care lucrați dvs.
- 17 Enumerați protocoalele utilizate în arhitectura *ARPANET*. Sunt oare aplicate aceste protocoale în rețeaua cu care lucrați dvs.?
- 18 Explicați termenul *Internetul obiectelor*.
- 19 EXPLOREAZĂ! În baza informațiilor publicate în Internet, aflați cum și în ce măsură *Internetul obiectelor* influențează procesele tehnologice și viața cotidiană a oamenilor.
- 20 CREEAZĂ! Elaborați un mic eseu care ar pune în evidență oportunitățile oferite de *Internetul obiectelor* și modalitățile de depășire a obstacolelor cu care el se confruntă.

7.4. Rețeaua Internet

Rețeaua globală *Internet* se bazează pe o topologie distribuită și include calculatoare separate, subrețele locale, regionale sau globale (fig. 7.5).

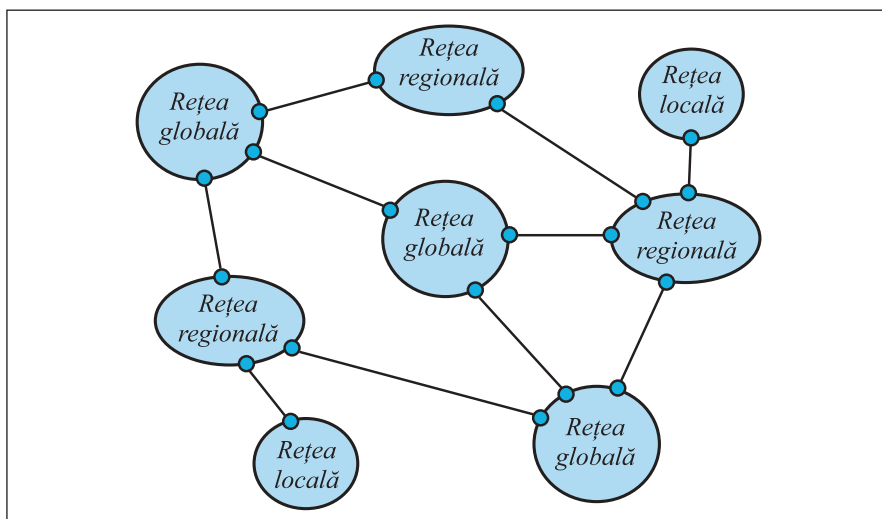


Fig. 7.5. Topologia Internetului

Interconectarea rețelelor se realizează cu ajutorul unor echipamente dedicate de rețea, denumite porți sau rutere. **Poarta** (*gateway*) este un calculator specializat destinat interconectării a două rețele ce utilizează protocoale total diferite de comunicație. **Ruterul** (*router*) este un calculator dedicat care interconectează rețelele ce folosesc protocoale identice și sunt utilizate pentru a determina cea mai bună cale de transmitere a pachetelor. Calculatoarele conectate la *Internet* se numesc **gazde** (*host*). Funcționarea rețelei este reglementată de circa 100 de protocoale.

Identificarea calculatoarelor în cadrul rețelei se face cu ajutorul **adreselor** *Internet*. Acestea pot fi de două tipuri: adrese numerice și adrese simbolice.

O **adresă numerică** este formată din 32 de cifre binare (4 octeți) și are structura prezentată în *figura 7.6*.

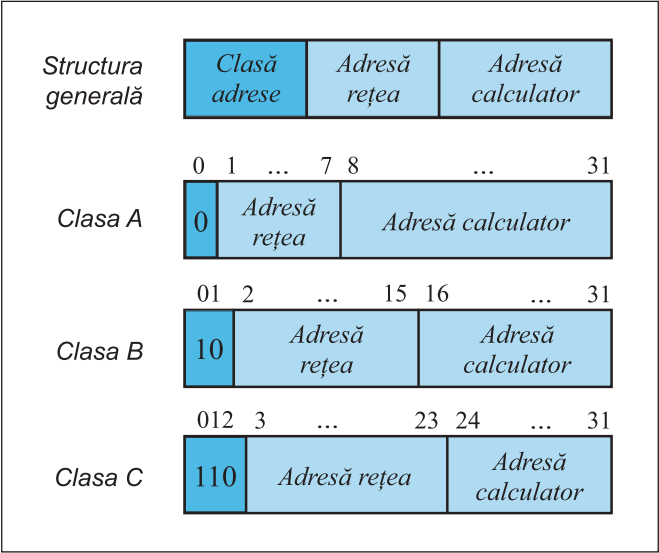


Fig. 7.6. Structura adreselor numerice

Întrucât *Internetul* este o „rețea de rețele”, adresa numerică conține adresa subrețelei (câmpul *Adresă Rețea*) și adresa calculatorului în cadrul subrețelei (câmpul *Adresă Calculator*). În funcție de numărul maxim de calculatoare pe care le poate identifica în cadrul unei subrețele, adresele se împart în clasele *A*, *B* și *C*. Adresele din clasele *D* și *E* au o destinație specială. Caracteristica adreselor *Internet* este prezentată în *tabelul 7.1*.

Tabelul 7.1

Caracteristica adreselor numerice

Clasa	Numărul de adrese disponibile	
	de rețea	de calculatoare
A	2 ⁷ =128	2 ²⁴ = 16 777 216
B	2 ¹⁴ =16 384	2 ¹⁶ = 65 536
C	2 ²¹ =2 097 152	2 ⁸ = 256

Adresele din clasa *A* sunt distribuite rețelelor mari, în special rețelelor globale. O astfel de rețea poate include circa 16 milioane de calculatoare. Adresele din clasa *B* sunt distribuite rețelelor medii, în special rețelelor regionale. O astfel de rețea poate include circa 65 mii de calculatoare. Adresele din clasa *C* sunt rezervate rețelelor relativ mici, care includ până la 256 de calculatoare. Fiecare adresă *Internet* este unică. Adresele sunt atribuite calculatoarelor de **Centrul Informațional al Rețelei** (*Network Information Center*).

De exemplu, numărul binar

10010010 00110011 00001001 11110111

este o adresă de clasa *B* și identifică calculatorul numărul

00001001 11110111

din cadrul rețelei

010010 00110011.

Pentru comoditate, adresele binare se exprimă în formă zecimală, transformând separat fiecare octet. Numerele zecimale corespunzătoare fiecărui octet se delimitează prin puncte.

În cazul exemplului de mai sus obținem:

$(10010010)_2 = (146)_{10};$

$(00110011)_2 = (51)_{10};$

$(00001001)_2 = (9)_{10};$

$(11110111)_2 = (247)_{10}.$

Prin urmare, în formă zecimală adresa respectivă va fi 146.51.9.247.

Adresele în formă zecimală și cu atât mai mult cele în formă binară sunt incomode pentru publicul larg. Din acest motiv, mai frecvent sunt utilizate adresele simbolice.

O **adresă simbolică** este formată din numele calculatorului-gazdă și nume de domenii separate prin puncte. **Domeniul** reprezintă un grup de calculatoare organizate tematic sau geografic. Orice domeniu poate fi împărțit în subdomenii, ajungându-se astfel la o structură ierarhică. Numele de domenii se indică în ordinea creșterii ariei de cuprindere.

De exemplu, adresele simbolice

c1.lme.ch.md

c5.lme.ch.md

specifică calculatoarele c1 și c5 din domeniul lme (Liceul „Mihai Eminescu”).

Adresele simbolice

c1.lic.ch.md

c9.lic.ch.md

specifică calculatoarele c1 și c9 din domeniul lic (Liceul „Ion Creangă”). Domeniile lme și lic sunt subdomenii ale domeniului ch (Chișinău). La rândul său, ch este un subdomeniu al domeniului md (Republica Moldova).

În mod similar, adresele simbolice

rector.ase.men.ro

decan.ase.men.ro

specifică calculatoarele rector și decan din domeniul ase (Academia de Studii Economice). Domeniul ase este un subdomeniu al domeniului men (Ministerul Educației Naționale), iar men este un subdomeniu al domeniului ro (România).

În mod normal, domeniul de cel mai înalt nivel este țara (md, ro, us etc.) sau tipul instituției (com – comercială, mil – militară, edu – de educație etc.).

Relațiile de incluziune între domenii pot fi reprezentate cu ajutorul diagramelor *Euler*, frecvent utilizate în teoria mulțimilor. Pentru exemplificare, în figura 7.7 sunt prezentate astfel de diagrame pentru unele adrese simbolice din domeniul md.

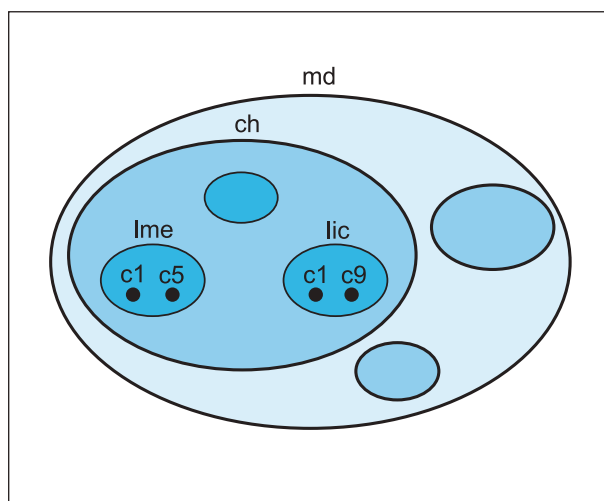


Fig. 7.7. Diagrame Euler pentru adrese simbolice

Pentru a **transforma** adresele simbolice în adrese numerice și invers, în fiecare domeniu există un **server de nume** (*name server*). Acest program gestionează domeniul respectiv fără intervenția serverelor ierarhic superioare. Prin urmare, în *Internet* nu există un calculator central care ar fi unicul responsabil de cele aproximativ 10^{10} adrese de rețea.

În cazul exemplului de mai sus (fig. 7.7), adresele calculatoarelor c1 și c5 sunt procesate în serverul lme, iar adresele calculatoarelor c1 și c9 – de serverul lic. Serverul ch va lucra numai cu adresele serverelor lic și lme, fără să se implice în procesarea adreselor de calculatoare din aceste domenii. În mod similar, serverul men nu procesează adresele calculatoarelor rector și decan, lăsând acest lucru în obligația serverului ase.

Întrebări și exerciții

- ❶ Desenați topologia rețelei *Internet*. Cum se interconectează subrețelele în cadrul *Internetului*?
- ❷ Care este destinația unei porți? Dar a unui ruter?
- ❸ Cum se identifică calculatoarele în *Internet*? Care sunt avantajele și dezavantajele adreselor numerice? Dar ale adreselor simbolice?
- ❹ Care sunt clasele de adrese numerice? Pentru ce este necesară o astfel de clasificare?
- ❺ Explicați destinația câmpurilor *Adresă rețea* și *Adresă calculator* ale unei adrese numerice.
- ❻ Cine atribuie adrese calculatoarelor din *Internet*?
- ❼ Determinați clasele următoarelor adrese. Precizați adresa subrețelei și adresa calculatorului în subrețea.

- | | |
|--------------------|--------------------|
| a) 45.201.19.63; | d) 192.109.58.170; |
| b) 201.165.213.91; | e) 15.21.207.250; |
| c) 154.36.79.200; | f) 217.15.69.113. |

- ❽ Care sunt criteriile de grupare a calculatoarelor în domenii? Numiți domeniile de cel mai înalt nivel.
- ❾ Sunt date următoarele adrese simbolice:

- | | |
|------------------------|---------------------|
| a) c1.lme.ch.md; | f) c4.lme.ch.md; |
| b) c3.lme.ch.md; | g) c5.lme.ch.md; |
| c) c1.lic.ch.md; | h) c9.lic.ch.md; |
| d) director.lic.ch.md; | i) prof.lic.ch.md; |
| e) elev1.lic.ch.md; | j) elev4.lic.ch.md. |

Precizați domeniile de calculatoare și relațiile de incluziune între domenii. Desenați diagramele *Euler* pentru adresele în cauză.

- ❿ Desenați o diagramă *Euler* pentru adresele simbolice ce urmează. Precizați domeniile de calculatoare și relațiile respective de incluziune.

- | | |
|-----------------------------|------------------------|
| a) rector.ase.men.ro; | d) rector.ase.met.md; |
| b) decan.ase.men.ro; | e) decan.ase.met.md; |
| c) student.info.ase.men.ro; | f) student.cib.met.md. |

- ⓫ Care este destinația unui server de nume? Ce adrese sunt procesate de un astfel de server?
- ⓬ Precizați serverele de nume pentru adresele din exercițiile 9 și 10. Indicați adresele pe care le procesează fiecare server.
- ⓭ EXPERIMENTEAZĂ! Aflați adresa numerică și adresa simbolică ale calculatorului cu care lucrați dvs. Precizați clasa de adresă, adresa subrețelei și adresa calculatorului în cadrul subrețelei. Desenați o diagramă *Euler* ce reprezintă relațiile de incluziune între domeniile la care aparține calculatorul dvs.

7.5. Servicii Internet

Rețeaua *Internet* oferă următoarea gamă de servicii:

- accesul calculatoarelor la distanță;
- transferul de fișiere;
- poșta electronică;
- știri și discuții;
- prezentarea și căutarea informațiilor etc.

Cooperarea calculatoarelor și programelor care oferă aceste servicii se bazează pe modelul client-server. De obicei, pe calculatorul beneficiarului de serviciu rulează programul client, iar pe calculatorul furnizorului de servicii rulează programul server.

Serviciul Telnet permite utilizatorului să aibă acces la calculatoarele aflate la distanță. După stabilirea conexiunii, calculatorul utilizatorului devine un simplu terminal al calculatorului aflat la distanță. În continuare, utilizatorul poate lansa în execuție pe calculatorul respectiv diverse programe, poate vizualiza fișiere, schimba directoare etc. Protecția calculatoarelor și a datelor respective se asigură prin utilizarea parolelor. Serviciul Telnet se utilizează pentru folosirea în comun a unor resurse foarte scumpe, de exemplu, a supercalculatoarelor.

Serviciul transfer de fișiere sau, mai scurt, **serviciul FTP** (*File Transfer Protocol*) permite utilizatorului să copie fișiere de pe calculatoare situate în diverse puncte geografice. Acest serviciu oferă două moduri de transfer al fișierelor:

- modul binar, în care se păstrează secvența de biți a fișierului, astfel încât originalul și copia sunt identice bit cu bit;
- modul text, în care se transferă seturi de caractere în codul *ASCII*.

În general, pentru a avea acces la serverul FTP, clientul trebuie să introducă o parolă. Totuși există servere publice (*FTP anonymous*) care permit accesul la fișiere fără a fi nevoie de o parolă specială.

Serviciul de poșta electronică (*electronic mail* sau, prescurtat, *e-mail*) a copiat modul de funcționare a poștei obișnuite.

Scrisoarea electronică, denumită **mesaj** (*message*) include:

- adresa destinatarului;
- subiectul, exprimat în câteva cuvinte;
- adresa expeditorului;
- textul scrisorii;
- fișiere atașate opțional.

Fișierele atașate pot fi de orice natură: texte, imagini, programe etc.

Scrisorile sunt depuse în fișiere speciale, denumite **cutii poștale** (*mail box*). Adresa unei cutii poștale are forma:

```
<nume cutie>@<Adresă calculator>,
```

unde:

<nume cutie> este denumirea cutiei poștale, de obicei acesta este numele de familie al utilizatorului sau o abreviere;

@ – simbolul „at” (la);

<Adresă calculator> – adresa simbolică a calculatorului client pe care este creată cutia poștală.

Exemple:

- 1) petrescu@c1.lme.ch.md
- 2) florea@director.lic.ch.md
- 3) ionescu@c1.lme.ch.md
- 4) barbu@director.lic.ch.md

Cititorii pot expedia scrisori autorilor acestui manual la adresa:

Anatol_Gremalschi@yahoo.com

Mesajele sunt transmise prin rețea de serverele de poștă care au rolul oficiilor și centrelor poștale tradiționale.

Serviciul de poștă electronică este foarte popular datorită avantajelor sale incontestabile, și anume: viteză, posibilitatea de a atașa la scrisori fișiere de orice natură, facilități avansate de redactare.

Cel mai modern serviciu de prezentare și căutare a informațiilor în *Internet* este **serviciul WWW** (*World Wide Web* – Pânza Mondială de Păianjen). În acest serviciu informația este prezentată în formă de pagini Web.

Pagina Web este un fișier scris în limbajul HTML (*Hypertext Markup Language* – Limbaj pentru marcarea hipertextului) și poate conține, în afară de informații propriu-zise, referințe la alte pagini Web. Paginile referite se pot afla pe același calculator sau pe calculatoare situate în diverse puncte geografice (fig. 7.8).

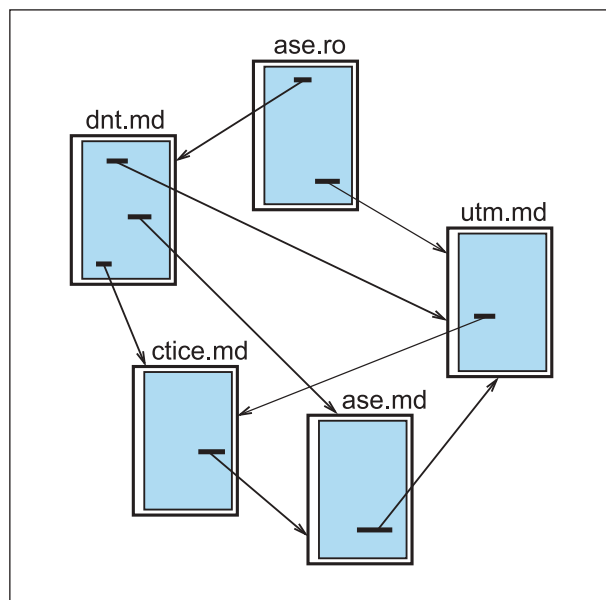


Fig. 7.8. Pagini Web

Tehnologia de cooperare în cadrul serviciului WWW este de tipul client-server. Utilizatorul care dorește să ofere publicului larg anumite informații instalează pe calculatorul său un program server și elaborează una sau mai multe pagini Web. Serverul interceptează cererile sosite de la alte calculatoare și asigură accesul la paginile respective. Calculatorul pe care sunt instalate paginile Web și serverul WWW se numește **site** (*site* – sediu, reședință).

Programul client asigură transferul și afișarea pe ecran a paginilor Web citite de pe diverse calculatoare din *Internet*. Imediat după pornire, el afișează propria pagină de referință (*home page* – pagina de acasă) și așteaptă indicațiile utilizatorului. Când utilizatorul activează o referință, programul client stabilește o conexiune cu serverul Web și copiază de la el pagina specificată în referință. Pagina copiată este afișată pe ecran.

În continuare, utilizatorul activează o altă referință, programul client stabilește din nou conexiunea cu un calculator din rețea, din nou se va citi o pagină Web etc. Cu alte cuvinte, utilizatorul „răsfoiește” paginile Web de pe diverse calculatoare, indiferent de poziția lor geografică. Din acest motiv programele client sunt numite **programe de răsfoire** sau **programe de explorare** (în engleză *browser* sau *explorer*).

În cadrul serviciului WWW resursele rețelei se specifică cu ajutorul unor adrese speciale, denumite adrese URL (*Uniform Resource Locator* – Locator Uniform de Resurse). Aceste adrese au forma:

```
<protocol>: // <Adresă simbolică>[:<port>]/<cale>/<fișier>
```

unde:

<protocol> – specifică denumirea protocolului pentru transferul datelor prin rețea;

<Adresă simbolică> – adresa calculatorului ce conține fișierul respectiv;

<port> – portul de acces (opțional);

<cale>/<fișier> – specificarea fișierului.

Pentru exemplificare, prezentăm câteva adrese URL ce conțin informații interesante:

<http://www.mecc.gov.md> – site-ul Ministerului Educației, Culturii și Cercetării, Republica Moldova;

<http://www.ctice.gov.md> – site-ul Centrului de Tehnologii Informaționale și Comunicaționale în Educație, Republica Moldova;

<https://www.sciencemuseum.org.uk> – site-ul Muzeului Științei din Londra, Regatul Unit al Marii Britanii și Irlandei de Nord;

<http://www.nasa.gov> – site-ul agenției NASA, Statele Unite ale Americii.

Amintim că notația [http](http://) specifică protocolul de transfer al hypertextelor (*Hypertext Transfer Protocol*).

În prezent, numărul fișierelor din rețeaua *Internet* este de ordinul miliardelor. Evident, nici nu poate fi vorba de căutarea informației necesare prin citirea individuală a fiecărui fișier. Pentru a simplifica căutarea informației, în cadrul rețelei *Internet* au fost create servere de căutare (*search engine*).

Serverul de căutare este un calculator puternic care explorează încontinuu rețeaua și citește paginile Web sau alte informații prezentate publicului larg.

Acestea sunt clasificate în funcție de datele pe care le conțin, iar adresele lor sunt reținute în baza de date de pe server.

Programul client adresează serverului de căutare o cerere în care indică de ce fel de informații are nevoie. Serverul interoghează baza de date și transmite clientului o listă de adrese la care pot fi găsite informațiile cerute.

Pentru exemplificare, amintim serverele de căutare frecvent utilizate:

<http://www.yahoo.com> – serverul YAHOO (*Yet Another Hierarchically Organized Oracle* – încă un oracol organizat ierarhic) al companiei *Yahoo! Inc.*;

<http://www.google.com> – serverul GOOGLE al corporației *Google Inc.*;

<http://www.bing.com> – serverul BING al corporației *Microsoft Inc.*;

<http://www.yandex.ru> – serverul YANDEX al companiei «Яндекс»;

<http://www.infoseek.com> – serverul Infoseek al firmei *Infoseek Corp.*

Accesul la aceste servere este gratuit.

Întrebări și exerciții

- ❶ Care este gama de servicii oferite de *Internet*? Cum colaborează calculatoarele din rețea în procesul prestării unui serviciu?
- ❷ Care este destinația serviciului *FTP*? Ce fișiere pot fi transferate prin acest serviciu?
- ❸ Studiați programul *FTP* instalat pe calculatorul dvs. Copiați câteva fișiere de pe serverele *FTP anonymous* sau de pe alte servere la care aveți acces.
- ❹ De ce depinde viteza de transfer a fișierelor? Determinați viteza de transfer a câtorva fișiere de pe serverele *FTP* aflate în Republica Moldova, România și Statele Unite ale Americii.
- ❺ Cum se specifică adresele în cadrul serviciilor de poștă electronică?
- ❻ Explicați cum interacționează calculatoarele client și serverele de poștă în cadrul serviciului respectiv.
- ❼ Ce informații conține o scrisoare electronică? Care dintre ele sunt opționale?
- ❽ Explicați cum se formează o adresă poștală. Aflați adresele poștale ale prietenilor dvs.
- ❾ Studiați programul de poștă electronică instalat pe calculatorul la care lucrați dvs. Verificați dacă acest program oferă următoarele facilități:
 - utilizarea mai multor cutii poștale;
 - clasificarea și păstrarea scrisorilor în dosare;
 - elaborarea scrisorilor conform unor șabloane;
 - cifrarea și descifrarea corespondenței;
 - semnalarea momentelor când a sosit corespondența;
 - verificarea faptului că scrisorile expediate au ajuns la destinație;
 - utilizarea semnăturilor electronice.
- ❿ Care sunt avantajele poștei electronice? Poate înlocui acest serviciu poșta tradițională?
- ⓫ LUCRUL ÎN GRUP. Formați grupe de câte patru utilizatori ai poștei electronice și determinați experimental viteza cu care se transmite corespondența.
- ⓬ Ce informații conține paginaWeb? Cum formează aceste pagini o „pânză de păianjen”?

- 13 Care este destinația unui server Web? Dar a unui client Web? Cum interacționează aceste programe?
- 14 Explicați modul de funcționare a unui client Web. Cum găsește acest program paginile Web amplasate pe diferite calculatoare?
- 15 Explicați modul de specificare a resurselor în *Internet* cu ajutorul adreselor URL. Care este semnificația câmpurilor acestor adrese?
- 16 EXPERIMENTEAZĂ! Determinați tipul programului de explorare a *Internetului* instalat pe calculatorul cu care lucrați dvs. Ce facilități oferă acest program? Citiți paginile Web adresele cărora sunt indicate în acest paragraf.
- 17 Care este destinația unui server de căutare? Ce servicii oferă un astfel de server?
- 18 EXPLOREAZĂ! Găsiți cu ajutorul unui server de căutare furnizorii de servicii *Internet* din Republica Moldova.
- 19 EXPLOREAZĂ! În afară de serviciile studiate în acest paragraf, rețeaua *Internet* oferă și alte servicii, cum ar fi: *Archie*, *Gopher*, *WAIS*, buletine de știri, discuții etc. Folosind un server de căutare, aflați informația respectivă despre aceste servicii.
- 20 PROIECTE! Elaborați paginile Web ale clasei și ale liceului dvs.

Capitolul 8

MODULE LA ALEGERE

Modulele din acest capitol sunt opționale. Ce înseamnă acest lucru? Aveți posibilitatea să alegeți unul dintre ele și să-l studiați în ritm propriu, de sine stătător sau împreună cu colegii, în procesul exersării și elaborării de proiecte, cu ajutorul mijloacelor de instruire asistată de calculator. După ce ați ales un modül, împreună cu profesorul de informatică veți decide ce programe de aplicații veți utiliza, veți avea grijă ca ele să fie licențiate sau cu distribuție liberă, vă veți asigura că ele sunt instalate pe calculatoarele din laboratoarele de informatică și pe cele personale, după caz.

Proiectele care se referă la subiecte ample, de exemplu, la orașe, sate, școli, vor fi elaborate în cadrul echipelor de elevi, fiecare membru al echipei specializându-se pe un anumit aspect, de exemplu, pe cel istoric, demografic, economic, cultural, etnografic etc.

Tot în echipe vor fi elaborate și proiectele ce se caracterizează printr-un bogat conținut multimedia, specializarea membrilor echipelor realizându-se după genul materialelor respective, de exemplu, imagini, secvențe audio, secvențe video, designul interfețelor grafice, programarea algoritmilor de prelucrare a datelor ș.a.

8.1. Introducere în rețele

Aceste tehnici vor fi studiate pe etape, după cum urmează.

1. Digitalizarea informației audio. Amintiți-vă din fizică caracteristicile sunetului ca fenomen fizic și parametrii ce influențează percepția de către om a undelor sonore. Din informatică amintiți-vă codificarea digitală a informației audio. Utilizând bogatele resurse ale Internetului, aflați cum se realizează compresia și decompresia datelor audio, care sunt formatele fișierelor ce conțin date audio.

Acordați o atenție deosebită echipamentelor destinate înregistrării digitale a sunetului și redării acestuia: parametrii tehnici, interacțiunea cu calculatoarele personale, costurile, domeniile de utilizare. În funcție de dotările laboratorului de informatică și de echipamentele digitale personale de care dispuneți, de exemplu, reportofoane digitale, plăci de sunet, microfoane, difuzoare, amplificatoare, studiouri digitale multimedia etc., decideți referitor la modalitățile de creare, stocare și redare a înregistrărilor sonore.

Exercițiile recomandate pentru această etapă sunt:

- explicarea principiilor de codificare și decodificare a datelor audio;
- determinarea volumului datelor audio necomprimate cunoscând parametrii de codificare;

- determinarea formatului audio cunoscând extensiunile denumirilor de fișiere;
- identificarea și explicarea parametrilor de bază și a caracteristicilor principale ale echipamentelor frecvent utilizate în colectarea, înregistrarea, prelucrarea și reproducerea secvențelor audio.

Pentru o înțelegere mai aprofundată a proceselor fizice și informatice legate de înregistrarea digitală și redarea sunetelor, se recomandă efectuarea următoarelor *studii de caz*:

- Înregistrarea analogică și înregistrarea digitală a sunetului.
- Analiza comparată a echipamentelor destinate prelucrărilor digitale audio.
- Analiza comparată a formatelor de fișiere audio.

Consolidarea și aprofundarea cunoștințelor în domeniul digitalizării informației audio poate fi făcută elaborând următoarele *proiecte de cercetare*:

- Influența frecvenței de discretizare a semnalelor sonore asupra calității de redare a acestora.
- Influența mărimii pasului de cuantificare a semnalelor sonore asupra calității de redare a acestora.
- Influența gradului de compresie a semnalelor sonore asupra calității de redare a acestora.

2. Prelucrarea înregistrărilor digitale audio. Studiați atent oferta de programe licențiate și cu distribuție liberă, destinate prelucrărilor digitale audio. Decideți ce program veți utiliza și instalați programul respectiv pe calculatorul personal cu care lucrați dumneavoastră.

Studiați mediul de lucru și facilitățile oferite de editorul digital audio cu care veți lucra. Acordați o atenție deosebită formatelor acceptate de fișiere audio, modalităților de stocare și organizare a fișierelor audio, compatibilității editorului cu echipamentele externe de înregistrare și redare a fișierelor audio. Convingeți-vă că interfețele grafice ale editorului digital audio sunt intuitive și ușor de utilizat.

Experimentând cu cele mai diverse înregistrări audio digitale, formați-vă următoarele abilități de prelucrare a acestora:

- secvențierea și concatenarea fragmentelor audio;
- transformări de amplitudine (amplificare, mixare normalizare);
- modificarea tonului și a duratei de redare;
- filtrarea semnalelor sonore;
- aplicarea efectelor.

În procesul prelucrării înregistrărilor digitale audio, utilizați facilitățile de analiză a sunetelor oferite de editoarele respective: analiza spectrului, modificarea volumului.

În scopul organizării raționale a fișierelor audio pe purtătorii de informație și al racordării fișierelor respective la cerințele înaintate față de calitatea redării sunetelor digitale, efectuați conversii de formate audio.

Exercițiile ce vor contribui la consolidarea cunoștințelor teoretice și la dezvoltarea abilităților practice de utilizare a echipamentelor digitale și a programelor destinate prelucrărilor audio includ:

- utilizarea facilităților oferite de editoarele digitale audio;
- analiza comparată a înregistrărilor uneia și aceleiași secvențe sonore efectuate în formate audio diferite;

- analiza calității de redare a secvențelor audio în funcție de parametrii de înregistrare și stocare;
- conversia formatelor fișierelor audio;
- transmiterea fișierelor audio prin diferite mijloace de comunicații digitale.

O sinergie a mijloacelor informatice și a celor specifice artelor audiovizuale poate fi asigurată prin elaborarea următoarelor proiecte:

- Elaborarea fundalului audio pentru evenimentele importante din viața personală, a familiei, a clasei, a școlii, a comunității.
- Înregistrarea și mixarea informațiilor audio ce provin de la mai multe surse pe durata unor evenimente festive, sportive, artistice, recreative.
- Elaborarea pistelor sonore pentru filmele video despre diversele evenimente din viața școlii și a comunității.

3. Digitalizarea informației video. Pentru început va trebui să vă amintiți din fizică natura ondulară și natura corpusculară a luminii și să țineți cont de faptul că în informatică lumina este tratată ca fiind o undă electromagnetică. În continuare, aflați din Internet modul în care ochiul uman percepe lumina și cum din mărimile parametrilor fizici ai undelor electromagnetice derivă culorile, luminanța și contrastul imaginilor. Pentru o înțelegere mai profundă a proceselor de digitalizare a informațiilor video, va trebui să vă împrăștiți cunoștințele despre metodele de cuantizare a imaginilor dinamice și de calculare a cantității de informație conținute de acestea. Acordați o atenție deosebită frecvenței de discretizare în timp, frecvenței de discretizare în spațiu, pasului de cuantificare a semnalului video, puterii de rezoluție, numărului de culori de bază.

Pornind de la faptul că foarte des imaginile video conțin informații ce sunt redundante în cazul percepției acestora de către om, faceți cunoștință cu metodele de compresie și decompresie a datelor video, cu formatele respective de fișiere. Identificați legătura dintre ratele de biți și calitatea redării informațiilor video, înregistrate în formă digitală.

Consolidați cunoștințele teoretice prin studierea echipamentelor destinate înregistrării digitale a informațiilor video și redării acestora: parametrii tehnici, interacțiunea cu calculatoarele personale, costurile, domeniile de utilizare. În funcție de dotările laboratorului de informatică și de echipamentele digitale personale de care dispuneți, de exemplu, camere video digitale, plăci video, obiective, vizualizatoare, proiectoare multimedia, studiouri digitale multimedia etc., decideți referitor la modalitățile de creare, stocare și redare a înregistrărilor digitale video.

Exercițiile recomandate pentru această etapă sunt:

- descrierea domeniilor de activitate în care se utilizează informația video;
- descrierea și compararea parametrilor undelor electromagnetice din perspectiva percepției lor de către ființele umane;
- descrierea și explicarea factorilor video ce pot periclita sănătatea;
- explicarea principiilor de codificare și decodificare a datelor video;
- determinarea volumului datelor video necomprimate cunoscând parametrii de codificare;
- determinarea formatului video cunoscând extensiunile denumirilor de fișiere;
- identificarea și explicarea parametrilor de bază și a caracteristicilor principale

ale echipamentelor frecvent utilizate în colectarea, înregistrarea, prelucrarea și reproducerea secvențelor video.

Studiile de caz, propuse pentru a fi efectuate de către elevi, sunt:

- Înregistrarea analogică și înregistrarea digitală a informațiilor video.
- Percepția diverselor unde electromagnetice de către ființa umană.
- Impactul modificării parametrilor undelor electromagnetice asupra percepției acestora de către ființa umană.

Proiectele de cercetare listate mai jos vor contribui la înțelegerea mai profundă a modului în care realizările teoretice ale informaticii influențează tehnologiile de colectare, înregistrare, stocare, prelucrare și redare a informațiilor video:

- modul în care variația frecvenței de discretizare a semnalelor video influențează calitatea de redare a acestora;
- modul în care variația pasului de cuantificare a semnalelor video influențează calitatea de redare a acestora;
- influența gradului de compresie a semnalelor video asupra calității acestora.

4. Prelucrări digitale video. Pe piața de produse soft și în distribuție liberă sunt promovate cele mai diverse programe de calculator, destinate prelucrării înregistrărilor video. Utilizatorii pot alege atât din produsele destinate profesioniștilor, cât și din cele destinate amatorilor. De obicei, sistemele de operare dispun de aplicații destinate redării înregistrărilor video, iar unele dintre ele – chiar și de editoare video.

Prin urmare, înainte de a parcurge această etapă, fiecare elev, sub îndrumarea profesorului de informatică, va decide ce produse-program va utiliza, ținând cont de faptul că ele trebuie să fie compatibile cu echipamentele digitale utilizate pentru înregistrarea și redarea informațiilor video. De asemenea, se va ține cont și de faptul că editoarele digitale video necesită resurse semnificative de calcul, atât de memorie (de ordinul Giga octeților), cât și de capacități de prelucrare (de ordinul Giga instrucțiuni pe secundă).

Studierea facilităților oferite de editoarele digitale video va avea drept scop formarea și dezvoltarea următoarelor abilități:

- secvențierea și concatenarea fragmentelor video;
- montarea fragmentelor video;
- transformări temporale;
- asocierea pistei sonore;
- modificarea duratei de redare;
- filtrarea semnalelor video;
- aplicarea efectelor;
- subtitrarea secvențelor video;
- conversia formatelor.

Exercițiile recomandate pentru această etapă sunt:

- utilizarea principalelor facilități ale unui editor digital video;
- analiza comparată a înregistrărilor uneia și aceleiași secvențe efectuate în formate video diferite;
- analiza calității de redare a secvențelor video în funcție de parametrii de înregistrare și stocare;
- conversia formatelor fișierelor video.

Tematica *proiectele recomandate* va fi racordată la cele propuse mai sus în cazul prelucrării digitale a secvențelor audio și va include:

- Elaborarea fundalului video pentru evenimentele importante din viața personală, a familiei, a clasei, a școlii, a comunității.
- Înregistrarea și mixarea informațiilor video ce provin de la mai multe surse pe durata unor evenimente festive, sportive, artistice, recreative.
- Montarea filmelor video despre evenimente din viața școlii și a comunității.

5. Diseminarea informațiilor multimedia. La această etapă, de sine stătător sau sub îndrumarea profesorului de informatică, studiați serviciile *online*, care vă oferă posibilitatea:

- să căutați resursele multimedia dorite;
- să diseminați propriile elaborări multimedia;
- să redactați secvențele multimedia în regim *online*.

Se recomandă să elaborați un catalog al serviciilor preferate și să studiați în detaliu fiecare dintre facilitățile serviciilor frecvent utilizate de publicul larg.

Exercițiile propuse în cadrul acestei etape sunt:

- transmiterea fișierelor multimedia prin mijloace de comunicații digitale;
- căutarea informațiilor multimedia pe Internet;
- înregistrarea și crearea propriilor profiluri în cadrul serviciilor online;
- publicarea propriilor elaborări multimedia;
- includerea în paginile Web, personale, a clasei, a școlii, a asociațiilor de elevi, a legăturilor către propriile resurse multimedia publicate online.

În calitate de *studiu de caz*, vă recomandăm să faceți o analiză comparată a serviciilor online de diseminare a informațiilor multimedia, identificând avantajele și dezavantajele acestora, domeniile de aplicare în funcție de specificul informațiilor multimedia destinate diseminării în spațiul virtual.

6. Etica digitală și respectarea dreptului de autor. Dacă în Antichitate textele inscripționate pe plăci de piatră sau scrise pe papyrus erau accesibile unui număr foarte mic de oameni, odată cu apariția tiparului, numărul oamenilor ce au început să aibă acces la cărți și, ulterior, la ziare, a crescut exponențial. Deși radioul și televiziunea au simplificat și mai mult procesele de difuzare a informațiilor, instituțiile respective, adică tipografiile, stațiile de emisie radio și TV aparțineau și mai aparțin fie statelor, fie marilor companii. În consecință, cetățenii de rând nu prea aveau posibilitatea să-și disemineze de sine stătător, pentru un număr cât mai mare de oameni, cărțile, fotografiile, fonogramele, filmele create de ei. Situația s-a schimbat radical odată cu apariția Internetului și a serviciilor *online* de difuzare a produselor multimedia, care permit fiecăruia dintre noi să supună atenției marelui public toate informațiile pe care le considerăm ca fiind importante, interesante, originale sau valoroase din punct de vedere științific sau artistic.

Indiscutabil, această libertate de exprimare contribuie la consolidarea și dezvoltarea societăților democratice, însă ea cere de la fiecare dintre noi respectarea în spațiul virtual a anumitor reguli. Ați studiat aceste reguli, denumite generic *reguli de etică digitală*, începând cu clasele de gimnaziu. Totuși, vă recomandăm cu fermitate să vă împrăștiți cunoștințele în acest domeniu, să respectați cu strictețe regulile respective și să nu uitați că într-o lume multiculturală și policonfesională cuvintele, imaginile, secvențele sonore și secvențele video pot fi

interpretate și înțelese în mod diferit. În consecință, înainte de a posta pe Internet un produs multimedia elaborat de dumneavoastră, aveți grijă ca el să transmită mesaje pozitive, să promoveze valorile general-umane, să încurajeze participarea și incluziunea.

Un alt aspect foarte important legat de diseminarea produselor multimedia constă în necesitatea respectării dreptului de autor și a drepturilor conexe. Amintim că dreptul de autor se extinde asupra operelor literare, artistice și științifice exprimate în următoarele forme:

- scrisă (manuscris, text dactilografiat, partitură etc.);
- orală (interpretare publică etc.);
- imprimare audio sau video (mecanică, magnetică, digitală, optică etc.);
- de imagine (desen, schiță, pictură, plan, poză etc.);
- tridimensională (sculptură, model, machetă, construcție etc.).

Prin urmare, înainte de a include în propriile elaborări unele fragmente din operele menționate mai sus, informați-vă dacă deținătorii dreptului de autor permit acest lucru.

De asemenea, nu prezentați drept creații personale ideile, fragmentele de text, imaginile, fonogramele, videogramele preluate din opere și produse multimedia create de alte persoane, întrucât acest fapt reprezintă un plagiat. Dacă autorii operelor și produselor multimedia respective permit preluarea de fragmente (acest lucru poate fi aflat consultând licențele respective), indicați în mod obligatoriu sursa: denumirea operei, editura care a publicat-o, referința URL ș.a.m.d.

Exercițiile recomandate pentru această etapă sunt:

- identificarea însemnelor ce declară dreptul de autor;
- explicarea regulilor ce vizează respectarea dreptului de autor;
- identificarea tipurilor licențelor de distribuție;
- utilizarea licențelor pentru distribuție.

În calitate de *studiu de caz* se recomandă analiza respectării drepturilor de autor în propriile produse multimedia, în produsele multimedia elaborate și publicate online de colegii de clasă și de școală, de către elevii din alte instituții de învățământ. Folosiți în acest scop facilitățile oferite de motoarele de căutare a informațiilor text, informațiilor grafice, informațiilor audio și video. De asemenea, vă vor fi utile și programele antiplagiat, disponibile *online* pe Internet.

8.2. Programarea vizuală

Istoric, primele programe au fost scrise în limbajul cod-calculator. Ulterior, s-a trecut la limbajele de asamblare și limbajele de programare de nivel înalt. În cazul limbajelor de programare de nivel înalt, programele reprezintă texte, care, pentru a fi „înțelese” de calculator, sunt „traduse” în coduri binare.

Pe parcursul a unei lungi perioade de timp, produsele-program au fost elaborate cu ajutorul limbajelor de programare de nivel înalt, folosind la început în acest scop simple editoare de texte, iar ulterior, medii integrate de dezvoltare a programelor. Aceste medii, pe lângă editarea textelor și introducerea rapidă de modificări, permit lansarea în execuție și depanarea programele aflate în curs de

elaborare. Cunoașteți deja astfel de medii de programare și chiar ați elaborat și depanat mai multe programe în ele.

Indiferent de mediile de dezvoltare a programelor utilizate fie în scopuri didactice, fie în scopuri de producție, în ele se lucrează cu texte. Însă, odată cu creșterea complexității programelor, textele respective au devenit voluminoase și greu percepute chiar și de autorii acestora, fapt ce a stimulat dezvoltarea așa-numitei *programări vizuale* – mijloc de elaborare a programelor nu prin scrierea de texte, ci prin manipularea cu obiecte grafice.

Cunoașteți deja mediul de programare vizuală *Scratch*, special elaborat pentru a studia informatica începând cu clasele primare. În acest mediu de programare elevii implementează algoritmi de comandă cu așa-numiții spiriduși prin manipularea cu obiecte grafice ce reprezintă spiridușii propriu-ziși, comenzile pe care ei trebuie să le execute și blocurile de control ale fluxului de execuție: ramificări în funcție de respectarea anumitor condiții, organizarea ciclurilor și subprogramelor.

Mijloace clasice de programare vizuală, elaborate în scopuri de producție, sunt aplicațiile de calcul tabelar, studiate în clasele gimnaziale. În aceste aplicații, utilizatorul implementează algoritmi de prelucrare a datelor numerice nu prin scrierea de programe într-un limbaj de nivel înalt, de exemplu, în PASCAL sau C++, ci prin operarea asupra unor obiecte grafice specifice, cum ar fi foile de calcul, celulele, rândurile, coloanele, diagramele, graficele, formularele etc.

În prezent, renumitele companii ce activează în domeniul tehnologiei informației și comunicațiilor propun o gamă foarte largă de mijloace de programare vizuală, mijloace ce ușurează foarte mult implementarea chiar și a celor mai complecși algoritmi. Utilizarea tot mai largă a programării vizuale deschide noi oportunități nu doar pentru persoanele ce doresc să facă o carieră profesională în informatică, ci și pentru cele pasionate de artele digitale, de instruirea asistată de calculator, de elaborarea de jocuri computerizate interactive.

Prin urmare, cunoașterea programării vizuale va permite elevilor să ia decizii argumentate referitoare la continuarea studiilor postliceale și la digitalizarea acelor domenii de activitate profesională în care ei au de gând să se afirme. Iar pentru a facilita procesul de studiere a programării vizuale, recomandăm elevilor să parcurgă următoarele etape:

1. Selectarea mediului de programare vizuală. Evident, orice program elaborat în mediile de programare vizuală are echivalentul său într-o formă de text, scrisă într-un limbaj de programare de nivel înalt. Mai mult ca atât, pentru a implementa eficient anumiți algoritmi de prelucrare a datelor, uneori chiar și în programarea vizuală, se cere și scrierea de subprograme într-un limbaj de nivel înalt. Prin urmare, la selectarea mediului de programare vizuală se va ține cont nu doar de ușurința de utilizare a acestuia, dar și de orientarea lui spre un anumit limbaj de programare de nivel înalt.

De exemplu, elevii care au studiat limbajele din familia PASCAL vor opta pentru mediile de programare vizuală *Delphi* sau *Lazarus*, iar cei ce au studiat limbajele din familia C – pentru mediile de programare vizuală *Microsoft Visual Studio* sau *Code::Blocks*.

Elevii care doresc să studieze doar conceptele de bază ale programării vizuale, fără a apela la un limbaj profesional de programare de nivel înalt, o pot face

utilizând *MIT App Inventor*, care folosește un limbaj vizual și interfețe grafice, similare celor din mediul de programare *Scratch*.

Recomandăm elevilor, înainte de a opta pentru un anumit mediu de programare vizuală, să se familiarizeze cu clasificarea acestora, cu parametrii lor tehnici și ergonomici, cu cerințele pe care produsele respective le înaintează față de calculatoarele ce vor fi utilizate în procesul de învățare și capacitatea de transmisie cerută de la conexiunile Internet. De asemenea, ei ar trebui să se conducă de scopul urmărit: să-și aprofundeze competențele în domeniul elaborării interfețelor grafice destinate aplicațiilor de consolă și aplicațiilor WEB sau să-și dezvolte noi competențe în domeniile aflate la intersecția tehnologiilor informației și a artelor: animații digitale, jocuri pe calculator, realitatea augmentată etc.

2. Studiarea noțiunilor și a principiilor de bază a programării vizuale. La această etapă va trebui să vă familiarizați cu noțiunile de obiect grafic, conexiuni și acțiuni, cu tipurile de obiecte grafice primare și metaobiecte, cu proprietățile obiectelor. O atenție deosebită trebuie acordată înțelegerii specificului programării orientate pe evenimente în context vizual: situații, acțiuni și rezultate.

Principalele *exerciții* recomandate pentru această etapă sunt:

- explicarea principiilor programării vizuale;
- explicarea noțiunilor de bază ale programării vizuale;
- clasificarea obiectelor/instrucțiunilor grafice;
- controlul evenimentelor, acțiunilor, stărilor.

La o înțelegere mai profundă a specificului programării vizuale vor contribui *următoarele studii de caz*:

- Instrucțiunile grafice versus instrucțiunile scrise.
- Flexibilitatea programării vizuale.
- Depanarea programelor: vizuală și procedurală.
- Depanarea programelor: vizuală și orientată pe obiecte.

3. Formarea și dezvoltarea abilităților de utilizare a mediilor de programare vizuală. Folosind în calitate de instrument de învățare sistemul de asistență al mediului vizual de programare selectat la prima etapă și instalat, după caz, pe calculator, elevii vor identifica componentele principale ale produsului-program utilizat și specificul interfeței grafice cu utilizatorul. Vor fi studiate în detaliu structura proiectului, tipologia fișierelor din componența proiectelor și modalitățile de organizare a acestora.

În baza proiectelor cu caracter didactic propuse de profesorul de informatică, elevii își vor forma abilitățile necesare pentru introducerea și editarea programelor, rularea și depanarea acestora. Principalele *exerciții* recomandate în acest scop sunt:

- identificarea componentelor principale ale mediului de programare vizuală;
- explicarea structurii proiectelor din cadrul mediilor de programare vizuală;
- identificarea fișierelor proiectelor;
- conectarea instrucțiunilor grafice în programe;
- depanarea programelor vizuale;
- derularea programelor vizuale.

La formarea unei viziuni profunde asupra bogatei diversități de mijloace destinate programării vizuale va contribui elaborarea următoarelor *studii de caz*:

- Avantajele și dezavantajele mediilor de programare vizuală.
- Analiza comparativă a mediilor de programare vizuale și tradiționale.
- Prezentarea unor medii și limbaje larg răspândite de programare vizuală (*Alice, Kodu, Scratch, ToonTalk, Cameleon, Filter Forge* ș.a.).

4. Formarea și dezvoltarea competenței de programare vizuală. Această etapă va începe cu studierea instrucțiunilor grafice și a criteriilor de clasificare a acestora. În continuare, elevii vor face cunoștință și vor studia în bază de mici exemple facilitățile ce asigură:

- Realizarea mișcării: deplasarea, rotirea, controlul apariției unui anumit eveniment, controlul apariției unei anumite situații.
- Setarea aspectului: redimensionarea, afișarea mesajelor, afișarea valorilor, modificarea obiectelor, setarea vizibilității, setarea straturilor.
- Controlul sunetului: setarea, start, stop, durata, aplicarea efectelor.
- Prelucrarea evenimentelor: pornirea/oprirea programelor, acțiunile de taste, acțiunile asupra butoanelor șoricelului, acțiunile asupra obiectelor grafice, controlul fluxurilor de mesaje.
- Controlul fluxurilor de execuție: instrucțiunile de selecție, instrucțiunile ciclice, pauze, opriri, clonări.
- Controlul situațiilor: setarea proprietăților obiectelor, introducerea datelor, setarea proprietăților unităților periferice, eșalonarea în timp.
- Crearea și apelarea subprogramelor.

Exercițiile recomandate pentru această etapă sunt:

- deplasarea și controlul obiectelor grafice;
- afișarea mesajelor;
- setarea proprietăților geometrice/ de culoare/ vizibilitate/ ale obiectelor grafice;
- integrarea componentelor sonore (multimedia) în programele vizuale;
- controlul obiectelor grafice dinamice cu ajutorul tastaturii și al șoricelului;
- crearea și apelarea subprogramelor.

Tot la această etapă va demara și *elaborarea de proiecte*, tematica recomandată a căreia include:

- Interacțiunea dintre două sau mai multe personaje (obiecte) vizuale.
- Redarea dialogurilor între personaje.
- Jocurile elementare (prototenis, prinderea obiectelor căzătoare, urmăriri).
- Animarea poveștilor.

Elevii sunt încurajați să concretizeze de sine stătător sau sub îndrumarea profesorului de informatică tema proiectului la care ar prefera să lucreze, să decidă asupra modului de organizare a lucrului la proiecte propuse de ei, individual sau în grup, și să elaboreze planuri de desfășurare a lucrărilor necesare pentru realizarea proiectelor respective.

5. Prelucrarea datelor externe. În spatele oricăror obiecte grafice se află anumite date: informații referitoare la forma și aspectul acestora, amplasarea lor în spațiu, traiectoria de deplasare în spațiul de lucru, mesajele recepționate de la alte obiecte etc. Rezultatele prelucrărilor efectuate asupra datelor respective determină comportamentul în timp și în spațiu a obiectelor, interacțiunile dintre ele și mediul extern.

Pentru a studia modalitățile de implementare a algoritmilor de prelucrare a datelor, elevii își vor îmbogăți cunoștințele referitoare la tipurile de date simple și la tipurile de date structurate, la operațiile ce pot fi efectuate asupra datelor respective. O atenție deosebită va fi acordată specificului conversiei de tipuri și operațiilor aritmetice, relaționale și logice ce pot fi efectuate asupra datelor anume în mediile vizuale de programare.

În continuare, recomandăm elevilor să studieze obiectele vizuale pentru introducerea și afișarea datelor, evenimentele și acțiunile destinate modificării datelor: situațiile apărute în procesul execuției programului, comenzile venite de la tastatură, de la șoricel, de la alte echipamente periferice.

Exercițiile ce vor contribui la dezvoltarea abilităților de prelucrare a datelor în programele vizuale sunt:

- declararea și utilizarea variabilelor ce aparțin la tipuri de date simple și structurate;
- utilizarea în cadrul programelor vizuale a operațiilor aritmetice, relaționale și logice;
- modificarea valorilor variabilelor prin acțiuni și reacții la evenimente (acționare tastelor și a butoanelor șoricelului, sosirea mesajelor de la alte obiecte grafice sau din exterior etc.);
- afișarea rezultatelor prin asocierea acestora unui obiect grafic;
- declararea și utilizarea variabilelor ce aparțin tipurilor de date structurate;
- căutarea informațiilor în datele structurate;
- modificarea structurii datelor.

O înțelegere mai aprofundată a specificului prelucrării datelor o poate asigura efectuarea unui *studiu de caz* în care s-ar analiza avantajele și dezavantajele modalităților de implementare a algoritmilor de calcul prin programare textuală și prin programare vizuală. Un astfel de studiu de caz ar permite și identificarea posibilităților de atașare a programelor textuale anumitor obiecte din componența programelor vizuale.

Proiectele recomandate pentru această etapă sunt:

- Calcularea numerelor maxime din mulțimile de numere enunțate de personaje (obiecte).
- Calcularea sumelor mulțimilor de numere enunțate de personaje.
- Găsirea numerelor prime din mulțimea celor enunțate de personaje.
- Calcularea celui mai mare divizor comun.
- Sortarea tablourilor.

În ansamblu, integrarea cunoștințelor și dezvoltarea abilităților de programare vizuală pot fi asigurate prin elaborarea unor proiecte complexe, cum ar fi interfețele grafice pentru aplicațiile de consolă sau aplicațiile Web, jocuri didactice, jocuri de agrement, programe pentru instruirea asistată de calculator, filme de animație (anime-uri), produse cu realitate augmentată (materiale publicitare, modele dinamice, reconstituiri în dinamică a obiectelor istorice ș.a.). Elaborarea unor astfel de proiecte este posibilă prin crearea unor echipe de elevi, în care, pe lângă o cooperare strânsă, trebuie să existe și o delimitare explicită a rolurilor și responsabilităților, o specializare în funcție de vocația fiecărui elev.

8.3. Limbaje de marcare a hipertextului

Posibil, în clasa precedentă ați studiat modulul opțional *Web design*. Studiind acest modúl, ați învățat cum să elaborați documente Web atât cu ajutorul aplicațiilor de uz general, de exemplu, cele de oficiu, cât și cu ajutorul aplicațiilor specializate, denumite editoare Web. În ambele cazuri, elevii, posibil, viitorii designeri Web, au operat cu obiectele paginilor Web aflate în curs de elaborare, adică cu texte, imagini, elemente de navigare, de introducere a informațiilor, de rulare a secvențelor audio și video etc.

Atât aplicațiile de uz general, cât și cele specializate, destinate elaborării documentelor Web, pe care le-ați studiat în clasele precedente, lucrează în regimul WYSIWYG (*What You See Is What You Get* – ceea ce vezi este anume ceea ce vei obține). În acest regim, dezvoltatorul Web operează în mod direct doar cu obiectele paginilor Web, fără a apela la posibilitățile oferite de limbajul HTML.

Amintim că limbajul HTML (*Hyper Text Markup Language* – Limbaj de Marcare a Hipertextului) a fost special inventat pentru a formata textele paginilor Web, a insera în ele elemente de control și referințe la cele mai diverse obiecte (alte pagini Web, imagini, secvențe audio, secvențe video etc.), aflate atât pe calculatorul local, cât și oriunde pe Internet.

Deși lucrul în regimul WYSIWYG este cu mult mai comod pentru dezvoltatorii de pagini Web decât cel de marcare a textelor cu ajutorul limbajului HTML, în multe cazuri „scrierea” nemijlocită a paginilor Web în limbajul HTML asigură o economie semnificativă de memorie și oferă designerului mai multe posibilități de așezare în pagină și de formatare a fiecăruia dintre obiectele din componența acesteia.

Se recomandă ca studierea limbajului HTML să se facă pe etape:

1. Elaborarea documentelor Web simple. La această etapă o atenție deosebită trebuie acordată înțelegerii conceptului limbajului HTML, rolului marcatorilor (etichetelor), modalităților de inserare a acestora în textele simple, neformatate, ce sunt afișate de programele de navigare ca pagini Web. Se va parcurge lista de marcatori, se va clarifica destinația fiecăruia dintre ei.

Crearea paginilor Web propriu-zise va începe cu elaborarea structurii generale a acestora, stabilirea obiectelor ce vor fi inserate în ele. La această etapă vor fi studiate în profunzime doar instrumentele HTML destinate formatării textelor:

- titluri;
- paragrafe;
- comentarii;
- stiluri fizice;
- stiluri logice;
- separatorii;
- liste ordonate;
- liste neordonate;
- liste de definiții.

Studierea instrumentelor de formatare a textelor din componența documentelor HTML se va face în strânsă legătură cu materiile referitoare la procesarea textelor studiate în clasele gimnaziale. Totodată, o atenție deosebită se va acorda cuvintelor-cheie din componența etichetelor, modalităților de organizare a etichetelor

în perechi, în cele de început și cele de sfârșit. Întrucât repertoriul de atribute ale etichetelor este unul foarte vast, eforturile de învățare vor fi concentrate nu pe memorarea acestora, ci pe formarea abilităților de utilizare a sistemelor de asistență *online* și *offline*, ce pot furniza informații detaliate despre fiecare etichetă și despre fiecare dintre atributele acesteia.

Activitățile practice recomandate pentru această etapă includ crearea paginilor Web care, pentru început, conțin doar texte formate în așa mod, încât să pună în evidență mesajele de transmis:

- Casa mea / Școala mea / Orașul meu / Satul natal;
- Clasa mea / Cercul meu școlar / Secția mea sportivă;
- Consiliul elevilor / Consiliul tineretului;
- Voluntariatul din școala mea, din satul / orașul meu;
- Librăria / Biblioteca / Casa de cultură / Salonul de muzică;
- Muzeul satului / orașului meu;
- Istoria satului / orașului meu;
- Oamenii remarcabili din satul / orașul meu;
- Arta în viața mea (pe domenii: muzica, arta plastică, arta decorativă);
- Sportul în viața mea;
- E minunat să fii sănătos;
- Săli de sport / de fitness;
- Saloane de înfrumusețare / Saloane de modă;
- Ateliere de cusut / de reparații (auto, calculatoare, electrocasnice, audio, video);
- Gospodăria țărănească;
- Magazine (mărfuri pentru copii, mărfuri școlare, de uz casnic, îmbrăcăminte).

Dacă în clasele precedente elevii au elaborat deja astfel de pagini, se recomandă afișarea fișierelor HTML în regimul *View Source* (Vezi Sursa) și analiza modalităților de utilizare a marcatorilor (etichetelor) de formatare a textelor din componența acestora.

2. Crearea legăturilor. Cel mai răspândit serviciu al Internetului – WWW (*World Wide Web* – Pânza Mondială [de Păianjen]), se bazează pe documente HTML, legate între ele, și fișiere, externe în raport cu documentele respective. Trecerea de la un document HTML la altul și afișarea/redarea de către programul de navigare a fișierelor externe se realizează cu ajutorul unor legături (referințe) speciale, denumite în ultimul timp și *linkuri*.

În cadrul unui document HTML, legăturile se creează asociind zonelor active ale paginii adrese URL. În calitate de zone active pot fi fragmentele de text sau de imagini, iar executarea unui clic pe o astfel de zonă impune programul de navigare să găsească resursa asociată pe calculatorul local sau în spațiul virtual, după caz, și să o afișeze/să o deruleze pe calculatorul local.

Se recomandă ca procesul de studiere a modalităților de creare a legăturilor să urmeze pașii:

- familiarizarea cu terminologia utilizată: ancoră, adresă URL, legătură internă, legătură externă, cale, comentariu la legătură;
- dezvoltarea capacităților de creare/de selectare a zonelor active ale documentelor HTML, formate atât din fragmente de text, cât și din imagini;
- finalizarea cu formatul marcatoarelor destinate creării legăturilor;

- crearea legăturilor către un document extern, aflat în același sau în alt catalog de pe calculatorul local;
- crearea legăturilor către un site;
- crearea legăturilor către o secvență a aceluiași sau a altui document HTML;
- crearea facilităților de lansare în execuție a unei aplicații de expediere a mesajelor;
- crearea facilităților de copiere a fișierelor.

La etapa inițială, în calitate de documente HTML în care vor fi create legături, se vor utiliza fișierele propuse de profesor, iar după formarea abilităților necesare se va trece la dezvoltarea acestora prin crearea de legături în documentele HTML, elaborate de elevi în cadrul proiectelor listate în activitățile practice ale etapei precedente (Casa mea, Clasa mea, Consiliul elevilor, Voluntariatul ș.a.).

3. Inserarea obiectelor multimedia. Inițial, paginile HTML, afișate pe ecran, conțineau doar texte. Ulterior, odată cu dezvoltarea tehnologiilor informației și a comunicațiilor, în ele au început să fie inserate legături la cele mai diverse obiecte multimedia: imagini statice, imagini animate, fotografii, secvențe sonore, secvențe video ș.a.

Procesul de studiere a modalităților de inserare în documentele HTML a legăturilor către obiectele multimedia va urma pașii:

- familiarizarea cu formatele fișierelor grafice, audio și video;
- organizarea fișierelor ce conțin obiecte multimedia în directoare, asociate cu documentele HTML în curs de elaborare;
- familiarizarea cu proprietățile obiectelor multimedia, relevante pentru inserarea lor în documentele HTML: dimensiuni, chenare, comentarii, controale, legături;
- studierea specificului obiectelor multimedia integrate: secvențe de control, atribute de integrare;
- inserarea în documentele HTML a imaginilor, secvențelor sonore sau video;
- setarea proprietăților obiectelor multimedia, inserate în documentele HTML.

La etapa inițială, formarea abilităților practice de inserare a obiectelor multimedia în documentele HTML se va realiza utilizând fișierele HTML și obiectele multimedia propuse de profesor. Ulterior, dezvoltarea acestor abilități se va realiza în cadrul proiectelor elaborate de elevi, tematica orientativă a cărora este indicată mai sus, în lista respectivă din prima etapă. În acest scop, elevii vor pregăti de sine stătător, în mod individual sau în echipă, acele obiecte multimedia (imagini, fotografii, secvențe sonore, secvențe video) ce contribuie semnificativ la formarea și punerea în evidență a mesajelor preconizate.

În procesul creării și/sau selectării obiectelor multimedia ce vor fi incluse în componența documentelor HTML se vor respecta cu strictețe etica digitală și dreptul de autor, în niciun caz nu se va recurge la plagiat.

4. Organizarea conținuturilor. Versiunile moderne ale limbajului HTML oferă mai multe facultăți de organizare a conținuturilor. În cazul învățământului liceal vom studia doar organizarea conținuturilor cu ajutorul tabelelor. În acest scop, recomandăm elevilor:

- să-și îmbogățească cunoștințele despre tabele: titlu, rând, coloană, celulă, titlu de rând, titlu de coloană, proprietățile celulelor, proprietățile conținuturilor din celule;

- să se familiarizeze cu marcatorii utilizați pentru crearea și formatarea tabelelor și cu atributele acestora;

- să organizeze conținuturile paginilor Web în curs de elaborare, utilizând așezarea în pagină cu ajutorul tabelelor;

- să ofere paginilor Web un design ce ar pune în evidență mesajele de transmis, editând și fremătând tablelele respective: combinarea și divizarea celulelor, aplicarea efectelor de culoare și de gradient, setarea proprietăților chenarelor și ale bordurilor.

Ca și în cazul etapelor precedente, inițial, marcatorii de tabel și atributele acestora vor fi studiați utilizând documente HTML foarte simple, propuse de profesor sau create de sine stătător de elevi, iar ulterior – în cadrul proiectelor elaborate de elevi în mod individual sau în echipă.

5. Testarea documentelor HTML. Pentru început, documentul HTML va fi testat local cu ajutorul unui program de navigare. În procesul testării se va verifica accesibilitatea tuturor obiectelor din componența documentelor HTML, corectitudinea referințelor, funcționarea instrumentelor de navigare, corectitudinea afișării imaginilor, corectitudinea redării secvențelor audio și video.

În cazul depistării unor erori, se va reveni la etapa de creare a paginilor Web și a obiectelor din componența acestora.

La finele studierii acestui modul, elevii sunt invitați să facă o analiză colectivă a documentelor HTML elaborate de ei, utilizând în acest scop atât criterii tehnice (corpurile de literă utilizate și mărimea acestora, așezarea în pagină, calitatea imaginilor, fidelitatea redării secvențelor audio și video, comoditatea navigării), cât și artistice (originalitatea, relevanța conținuturilor în raport cu mesajele de transmis, modul de punere în evidență a mesajelor, sortarea culorilor și a stilurilor de afișare).