MINISTERUL EDUCAȚIEI AL REPUBLICII MOLDOVA

Anatol GREMALSCHI
Iurie MOCANU
Ludmila GREMALSCHI

INFORMATICĂ

Manual pentru clasa a 10-a

Ştiința, 2012

CZU 004(075.3)
G 80

Elaborat conform curriculumului disciplinar în vigoare şi aprobat prin Ordinul ministrului educaţiei (nr. 211 din 11 aprilie 2012). Editat din sursele financiare ale *Fondului Special pentru Manuale*.

Comisia de experţi: *Teodora Gherman*, dr. în pedagogie, conferenţiar, şef Catedră Tehnologii Informaţionale Aplicate, Academia de Administrare Publică de pe lîngă Preşedintele Republicii Moldova; *Gheorghe Chistruga*, prof. şc., grad did. superior, Liceul Teoretic „Mihai Eminescu", Drochia; *Mihai Chigai*, prof. şc., grad did. I, Liceul Teoretic Caplani, rn. Ştefan-Vodă

Recenzenţi: *Gheorghe Ciocanu*, dr. habilitat în informatică, profesor universitar, Universitatea de Stat din Moldova; *Valeriu Cabac*, dr. în fizică şi matematică, conferenţiar universitar, Universitatea de Stat „Alecu Russo", Bălţi; *Mihai Şleahtiţchi*, dr. în psihologie şi pedagogie, conferenţiar universitar, Universitatea Liberă Internaţională din Moldova; *Tatiana Cartaleanu*, dr. în filologie, conferenţiar universitar, Universitatea Pedagogică de Stat „Ion Creangă", Chişinău; *Alexei Colîbneac*, Maestru în Arte, profesor universitar, Academia de Muzică, Teatru şi Arte Plastice, Chişinău

Redactor: *Vasile Bahnaru*
Corectori: *Mariana Belenciuc, Oana Stoian*
Redactor tehnic: *Nina Duduciuc*
Machetare computerizată: *Anatol Andriţchi*
Copertă: *Vitaliu Pogolşa*

Întreprinderea Editorial-Poligrafică *Ştiinţa*,
str. Academiei, nr. 3; MD-2028, Chişinău, Republica Moldova;
tel.: (+373 22) 73-96-16; fax: (+373 22) 73-96-27;
e-mail: prini@stiinta.asm.md

DIFUZARE:

ÎM Societatea de Distribuţie a Cărţii *PRO-NOI*
str. Alba-Iulia, nr. 23/1A; MD-2051, Chişinău;
tel.: (+373 22) 51-68-17, 51-57-49; fax: (+373 22) 50-15-81;
e-mail: info@pronoi.md; www.pronoi.md

Descrierea CIP a Camerei Naţionale a Cărţii
Gremalschi, Anatol
Informatică: Man. pentru clasa a 10-a / Anatol Gremalschi, Iurie Mocanu, Ludmila Gremalschi; Min. Educaţiei al Rep. Moldova. – Ch.: Î.E.P. *Ştiinţa*, 2012 (Tipogr. „SEREBIA" SRL). – 188 p.
ISBN 978-9975-67-818-6
004(075.3)

ISBN 978-9975-67-818-6

CUPRINS

INTRODUCERE

Impresionantele realizări în domeniul informaticii, crearea supercalculatoarelor şi a calculatoarelor personale, apariţia ciberspaţiului, a realităţii virtuale şi a Internetului presupun o cunoaştere profundă a principiilor de funcţionare şi a structurii calculatoarelor moderne. Din motive justificate, aceste principii au devenit o călăuză demnă de toată încrederea într-o lume aflată mereu în schimbare.

Manualul de faţă are drept scop însuşirea de către elevi a cunoştinţelor necesare pentru prelucrarea automată a informaţiei cu ajutorul calculatoarelor numerice.

Capitolul 1 include un amplu material teoretic şi practic referitor la definirea şi prelucrarea datelor structurate: a tablourilor, a şirurilor de caractere, a articolelor, a mulţimilor şi fişierelor. Sînt prezentate metodele de creare şi de prelucrare a fişierelor: asocierea fişierelor PASCAL cu fişiere externe, scrierea şi citirea componentelor unui fişier.

Capitolul 2 include expunerea unor cunoştinţe fundamentale din teoria informaţiei, şi anume: cantitatea de informaţie din mesajele continue şi discrete, codificarea şi decodificarea informaţiei, prezentarea informaţiei în calculator.

În **Capitolul 3** sînt expuse cunoştinţe fundamentale din domeniul aritmeticii sistemelor de calcul: sisteme de numeraţie şi operaţii aritmetice în sistemul binar, reprezentarea numerelor naturale, numerelor întregi şi a numerelor reale în calculator.

Capitolul 4 include unele cunoştinţe fundamentale din domeniul algebrei booleene. Sînt expuse noţiunile de variabilă, constantă şi funcţii booleene, se examinează funcţiile logice frecvent utilizate.

În **Capitolul 5** se studiază circuitele logice ale unui calculator numeric: sumatorul, comparatorul, codificatorul şi decodificatorul, bistabilul, registrul, numărătorul, generatorul de impulsuri.

Structura şi funcţionarea calculatorului sînt descrise în **Capitolul 6**. Materialul este expus în aşa mod încît structura unui calculator numeric poate fi înţeleasă şi însuşită metodic, de la porţi logice, dispozitive şi unităţi la sisteme de calcul. O atenţie deosebită se acordă interdependenţei dintre conceptele matematice şi realizarea fizică a echipamentelor unui sistem de calcul, interconexiunilor dintre resursele tehnice şi cele programate ale calculatorului.

În **Capitolul 7** se studiază reţelele de calculatoare. Sînt expuse tehnologiile de cooperare în reţea, topologia şi arhitectura reţelelor locale, regionale şi globale. Pentru a facilita explorarea ciberspaţiului, acest capitol include cunoştinţe fundamentale despre Internet şi serviciile din reţea: transferul fişierelor, poşta electronică, paginile Web.

Manualul este realizat în conformitate cu *Curriculumul disciplinar de informatică pentru învăţămîntul liceal*, aprobat prin Ordinul Ministerului Educaţiei al Republicii Moldova nr. 244 din 27 aprilie 2010. Repartizarea materialului pe profiluri – umanist şi real – este reflectată în cuprinsul manualului.

Capitolul 1

TIPURI DE DATE STRUCTURATE

1.1. Tipuri de date *tablou* (`array`)

Mulțimea de valori ale unui tip de date **`array`** este constituită din tablouri (tabele). Tablourile sînt formate dintr-un număr fixat de componente de acelaşi tip, denumit **tip de bază**. Referirea componentelor se face cu ajutorul unui **indice**.

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

```
type <Nume tip> = array [T1] of T2;
```

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.*

Fiecare componentă a unei variabile de tip *tablou* 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].
```

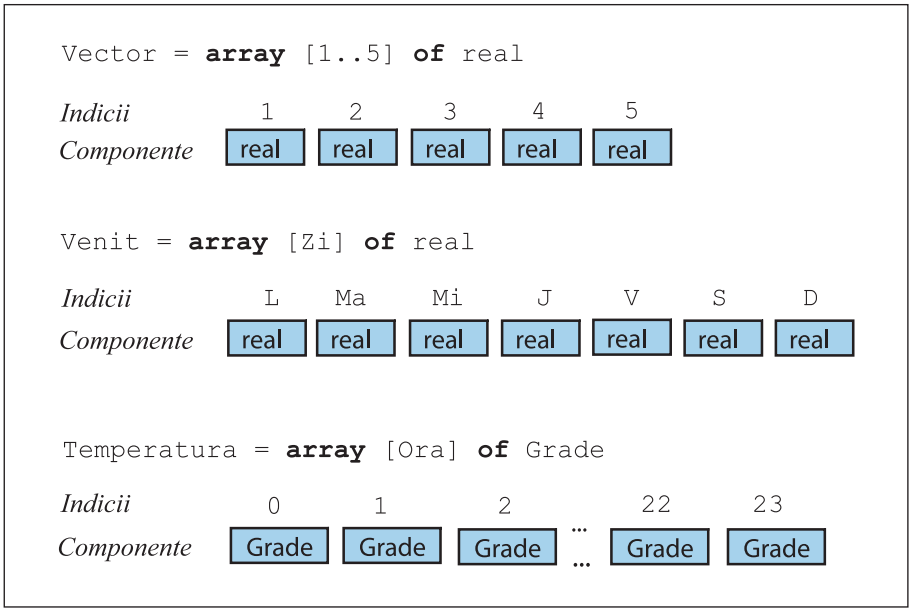


Fig. 1.1. Structura datelor de tip `Vector`, `Venit` şi `Temperatura`

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

```
Program P77;
{ Suma componentelor variabilei x de tip Vector }
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 `P77` poate fi modificat pentru a însuma n numere reale, $n \leq 100$:

```
Program P78;
{ Extinderea domeniului de aplicare a programului P77 }
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.
```

Tablourile bidimensionale se definesc cu ajutorul construcției

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

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

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

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

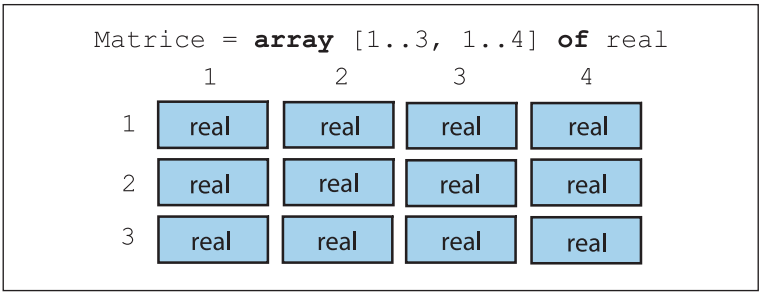


Fig. 1.2. Structura datelor de tip `Matrice`

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

De exemplu, în prezența declarației

```
var m : Matrice;
```

notația `m[1,1]` specifică componenta din linia 1, coloana 1 (vezi *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 afişează 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 P79;
{ 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 defineşte 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, întrucît optimizarea se efectuează în mod automat.

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

De exemplu, în prezența declarațiilor

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

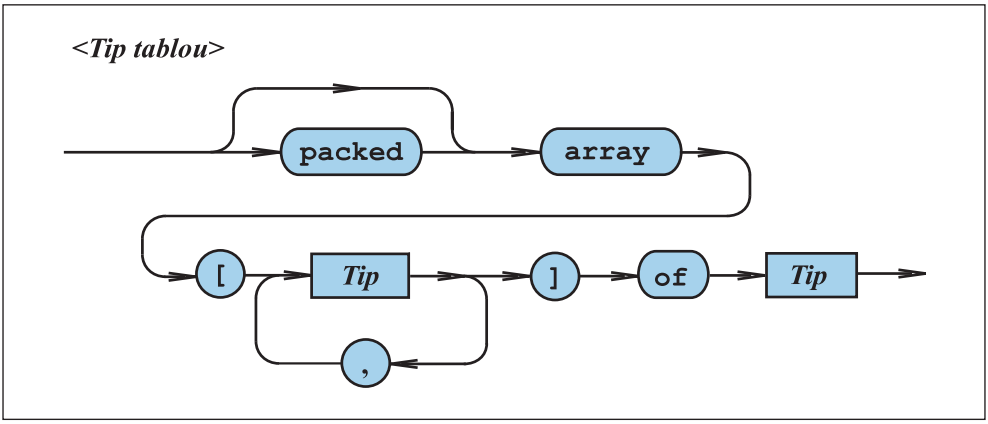


Fig. 1.3. Diagrama sintactică <*Tip tablou*>

instrucțiunea

```
a:=b
```

este corectă.

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 însuşi 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]` şi `T[3]` de tipul `Linie`. Variabila `L` este formată din 4 componente: `L[1]`, `L[2]`, `L[3]` şi `L[4]` de tipul `real`.

Prin urmare, atribuirile

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

sînt corecte.

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

Subliniem faptul că declarațiile de forma

array [T_1, T_2] **of** T_3

şi

array [T_1] **of array** [T_2] **of** T_3

definesc tipuri distincte de date. Prima declarație definește tablouri bidimensionale cu componente de tipul T_3. A doua declarație definește tablouri unidimensionale cu componente de tipul **array** [T_2] **of** T_3.

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

Întrebări şi exerciții

❶ Precizați tipul indicilor şi tipul componentelor din următoarele declarații:

```
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;
```

Reprezentați structura datelor de tipul `P`, `R`, `S` şi `T` pe un desen (*fig. 1.1* şi *1.2*).

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

❸ Scrieți formulele metalingvistice care corespund diagramei sintactice <*Tip tablou*> din *figura 1.3*.

❹ Se consideră declarațiile:

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

Scrieți expresia aritmetică a cărei valoare este:
a) suma primelor trei componente ale variabilei `x`;
b) suma tuturor componentelor variabilei `y`;
c) produsul tuturor componentelor variabilei `x`;
d) valoarea absolută a componentei a treia a variabilei `y`;
e) suma primelor componente ale variabilelor `x` şi `y`.

❺ Se consideră declarațiile

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

Componentele variabilei `v` reprezintă venitul zilnic al unei întreprinderi. Elaborați un program care:
a) calculează venitul săptămînal al întreprinderii;
b) calculează media venitului zilnic;
c) indică ziua în care s-a obținut cel mai mare venit;
d) indică ziua cu venitul cel mai mic.

❻ Se consideră declarațiile

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

Componentele variabilei `t` reprezintă temperaturile măsurate din oră în oră pe parcursul a 24 de ore. Elaborați un program care:
a) calculează temperatura medie;
b) indică maximul şi minimul temperaturii;
c) indică ora (orele) la care s-a înregistrat temperatura maximă;
d) indică ora (orele) la care s-a înregistrat temperatura minimă.

❼ Se consideră declarațiile

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

Componenta `C[r,z]` a variabilei `C` reprezintă consumul de energie electrică a oraşului `r` în ziua `z`. Elaborați un program care:
a) calculează energia electrică consumată de fiecare oraş pe parcursul unei săptămîni;
b) calculează energia electrică consumată zilnic de oraşele în studiu;
c) indică oraşul cu un consum săptămînal maxim;
d) indică oraşul cu un consum săptămînal minim;
e) indică ziua în care oraşele consumă cea mai multă energie electrică;
f) indică ziua în care oraşul consumă cea mai puțină energie electrică.

❽ Se consideră declarațiile

```
type Vector = array [1..5] of real;
     Matrice = array [1..3, 1..4] of real;
     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;
```

Care dintre atribuirile ce urmează sînt corecte?

a) `T[3]:=T[1];`
b) `M:=T;`
c) `L:=V;`
d) `L[3]:=x;`
e) `x:=i;`
f) `i:=x;`
g) `L[3]:=i;`
h) `i:=M[1,2];`
i) `x:=V[4];`
j) `L[3]:=V[4];`
k) `T[1]:=4;`
l) `T[2]:=V;`
m) `L:=T[3];`
n) `T[1,2]:=M[1,2];`

o) `T[1,2]:=M[1,2];`

p) `M[1]:=4;`

q) `M[1,3]:=L[2];`

r) `x:=T[1][2];`

s) `x:=M[1];`

t) `L:=M[1];`

u) `V[5]:=M[3,4];`

v) `L:=M[3,4].`

❾ Utilizînd un tip de date *tablou*, elaboraţi un program care realizează algoritmul lui Eratostene pentru calcularea numerelor prime mai mici decît un număr natural dat n ($n \leq 200$).

1.2. Tipuri de date *şir de caractere*

În limbajul-standard tipul de date *şir de caractere* reprezintă un caz special al tipului **array** şi se defineşte printr-o construcţie de forma

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

Mulţimea de valori ale tipului de date în studiu este formată din toate şirurile ce conţin exact n caractere.

Exemplu:

```
Program P80;
{ Siruri de caractere de lungime constanta }
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 afişat pe ecran:

```
Munteanu
Mihai
```

Întrucît şirurile de lungime diferită aparţin unor tipuri distincte de date, în cadrul programului `P80` nu sînt admise atribuiri de genul:

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

În astfel de cazuri, programatorul va completa datele respective cu spațiul pînă 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 afişată în totalitatea ei printr-un singur apel write(v) sau writeln(v).

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

Pentru a elimina acest neajuns, versiunile actuale ale limbajului permit utilizarea şirurilor de caractere de lungime variabilă. În Turbo PASCAL un tip de date **şir de caractere de lungime variabilă** se declară printr-o construcţie de forma:

```
type <Nume tip> = string;
```

sau

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

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

Asupra şirurilor de tip **string** se poate efectua operaţia de concatenare (juxtapunere), notată prin semnul „+”. Lungimea curentă a unei valori v de tip **string** poate fi aflată cu ajutorul funcţiei predefinite length(v) care returnează o valoare de tip integer. Indiferent de lungime, toate şirurile de caractere de tip **string** sînt compatibile.

Exemplu:

```
Program P81;
{ Siruri de caractere de lungime variabila }
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 afişate 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 şirurilor de caractere `N`, `P` şi `NP` se schimbă.

Asupra şirurilor de caractere sînt admise operaţiile relaţionale <, <=, =, >=, >, <>. Şirurile se compară componentă cu componentă de la stînga 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 acelaşi număr de componente sau de tip **`string`**. Evident, operanzii de tip **`string`** pot avea lungimi arbitrare.

De exemplu, rezultatul operaţiei

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

este `true`, iar rezultatul operaţiei

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

este `false`.

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

De exemplu, pentru `P='Mihai'` avem `P[1]='M'`, `P[2]='i'`, `P[3]='h'` ş.a.m.d. După executarea secvenţei de instrucţiuni

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

variabila `P` va avea valoarea `'Petru'`.

Programul ce urmează citeşte de la tastatură şiruri arbitrare de caractere şi afişează pe ecran numărul de spaţii în şirul respectiv. Derularea programului se termină după introducerea şirului *'Sfirsit'*.

```
Program P82;
{ Numarul de spatii intr-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='Sfirsit';
end.
```

Întrebări şi exerciţii

❶ Cum se defineşte un tip de date *şir de caractere*?

❷ Ce operaţii pot fi efectuate asupra şirurilor de caractere?

❸ Comentaţi următorul program:

```
Program P83;
{ Eroare }
var S : packed array [1..5] of char;
begin
  S:='12345';
  writeln(S);
  S:='Sfat';
  writeln(S);
end.
```

❹ Elaboraţi un program care:
a) determină numărul de apariţii ale caracterului `'A'` într-un şir;
b) substituie caracterul `'A'` prin caracterul `'*'`;
c) radiază din şir caracterul `'B'`;
d) determină numărul de apariţii ale silabei `'MA'` într-un şir;
e) substituie silabele `'MA'` prin silaba `'TA'`;
f) radiază din şir silaba `'TO'`.

❺ Precizaţi rezultatul operaţiilor relaţionale:

a) `'B' < 'A';`

b) `'BB' > 'AA';`

c) `'BAAAA' < 'AAAAA';`

d) `'CCCCD' > 'CCCCA';`

e) `'A A' = 'AA';`

f) `'BB ' < 'B B';`

g) `'A' = 'a';`

h) `'Aa' > 'aA';`

i) `'123' = '321';`

j) `'12345' > '12345'.`

❻ Se consideră şiruri de caractere formate din literele mari ale alfabetului latin şi spaţii. Elaboraţi un program care afişează şirurile în studiu după următoarele reguli:
– fiecare literă de la `'A'` pînă la `'Y'` se înlocuieşte prin următoarea literă din alfabet;
– fiecare literă `'Z'` se înlocuieşte prin litera `'A'`;
– fiecare spaţiu se înlocuieşte prin `'-'`.

❼ Elaboraţi un program care descifrează şirurile cifrate conform regulilor din exerciţiul 6.

❽ Se consideră m, $m \le 100$ şiruri de caractere formate din literele mici ale alfabetului latin. Elaboraţi un program care afişează pe ecran şirurile în studiu în ordine alfabetică.

❾ Şirul `S` este compus din cîteva propoziţii, fiecare terminîndu-se cu punct, semn de exclamare sau semnul întrebării. Elaboraţi un program care afişează pe ecran numărul de propoziţii din şirul în studiu.

1.3. Tipuri de date *articol* (`record`)

Mulţimea de valori ale unui tip de date **`record`** este constituită din articole (înregistrări). Articolele sînt formate din componente, denumite ***cîmpuri***. Spre deosebire de componentele unui tablou, cîmpurile pot fi de tipuri diferite. Fiecare cîmp are un nume (identificator de cîmp).

Un tip de date *articol* se defineşte printr-o structură de forma

```
type <Nume tip> = record
                      <Nume cîmp 1> : T1;
                      <Nume cîmp 2> : T2;
                      ...
                      <Nume cîmp n> : Tn;
                  end;
```

unde T_1, T_2, .., T_n specifică tipul cîmpurilor respective. Tipul unui nume de cîmp este arbitrar, astfel un cîmp poate să fie la rîndul 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; { virful A }
                   B : Punct; { virful B }
                   C : Punct; { virful C }
                end;
var T1, T2, T3 : Triunghi;
```

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

Fiind date două variabile de tip *articol* de acelaşi tip, numele variabilelor pot apărea într-o instrucţiune de atribuire. Această atribuire înseamnă copierea tuturor cîmpurilor din membrul drept în membrul stîng. De exemplu, pentru tipurile de date şi variabilele declarate mai sus instrucţiunile

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

sînt corecte.

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

Exemple:

1)
```
E1.Nume, E1.Prenume, E1.NotaMedie;
```

2)
```
E2.Nume, E2.Prenume, E2.NotaMedie;
```

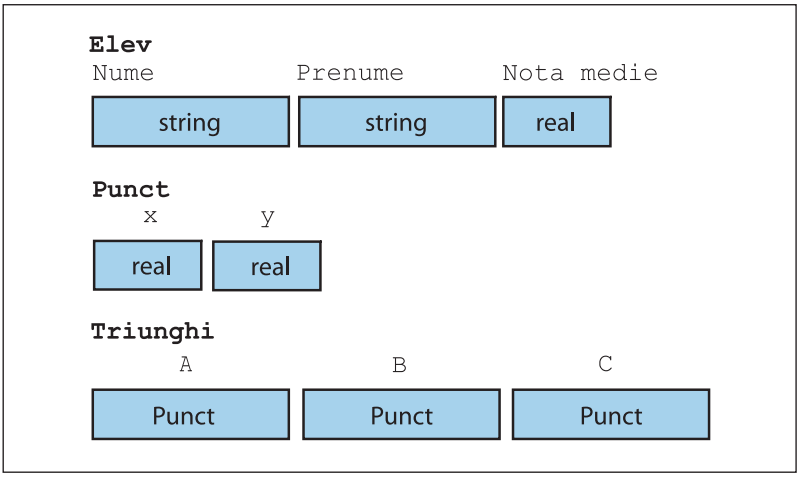


Fig. 1.4. Structura datelor de tip `Elev`, `Punct` şi `Triunghi`

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` ş.a.m.d.

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

```
Program P84;
{ 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 ş.a.m.d. Programul ce urmează citeşte de la tastatură datele referitoare la *n* elevi şi afişează pe ecran numele, prenumele şi nota medie a celui mai bun elev. Se consideră că elevii au note medii diferite.

```
Program P85;
{ 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 defineşte cu ajutorul diagramelor sintactice din *figura 1.5*. În completare la articolele cu un număr fix de cîmpuri, limbajul PASCAL permite utilizarea articolelor cu variante. Aceste tipuri de date se studiază în cursurile avansate de informatică.

Întrebări şi exerciții

❶ Care este mulțimea de valori ale unui tip de date *articol*?

❷ Indicați pe diagrama sintactică din *figura 1.5* drumurile care corespund definițiilor tipurilor de date *articol* din programele `P84` şi `P85`.

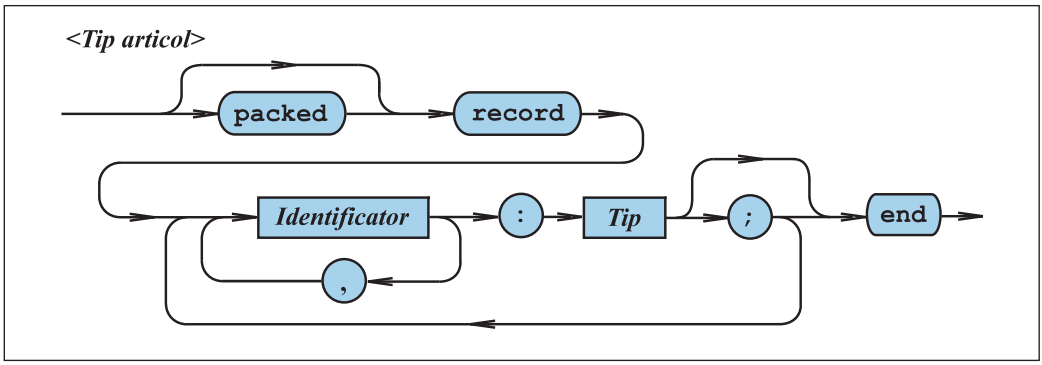


Fig. 1.5. Diagrama sintactică <*Tip articol*>

❸ Scrieți formulele metalingvistice pentru diagrama sintactică din *figura 1.5*.

❹ Se consideră următoarele tipuri de date

```
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;
```

Elaborați un program care citeşte de pe tastatură datele referitoare la n persoane ($n \leq 50$) şi afişează 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 vîrstă;
f) persoana cea mai tînără;
g) vîrsta 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 naşterii;
k) lista persoanelor de aceeaşi vîrstă (născuți în acelaşi an).

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

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

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

❻ Aria triunghiului este dată de formula lui Heron

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

unde *p* este semiperimetrul, iar *a, b* şi *c* sînt lungimile laturilor respective. Utilizînd tipurile de date `Punct` şi `Triunghi` din paragraful în studiu, elaboraţi un program care citeşte de la tastatură informaţiile referitoare la *n* triunghiuri ($n \leq 10$) şi afişează pe ecran:

a) aria fiecărui triunghi;
b) coordonatele vîrfurilor triunghiului cu aria maximă;
c) coordonatele vîrfurilor triunghiului cu aria minimă;
d) informaţiile referitoare la fiecare triunghi în ordinea creşterii ariilor.

1.4. Instrucţiunea `with`

Componentele unei variabile de tip *articol* se specifică explicit prin numele variabilei şi denumirile de cîmpuri, separate prin puncte.

De exemplu, în prezenţa declaraţiilor

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

componentele variabilei `A` se specifică prin `A.NumePrenume`, `A.ZileLucrate`, `A.PlataPeZi` şi `A.PlataPeLuna`.

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

Sintaxa instrucţiunii în studiu este:

<*Instrucţiune* **`with`**> ::= **`with`** <*Variabilă*> {, <*Variabilă*>} **`do`** <*Instrucţiune*>

Diagrama sintactică este prezentată în *figura 1.6*.

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

Exemplu:

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

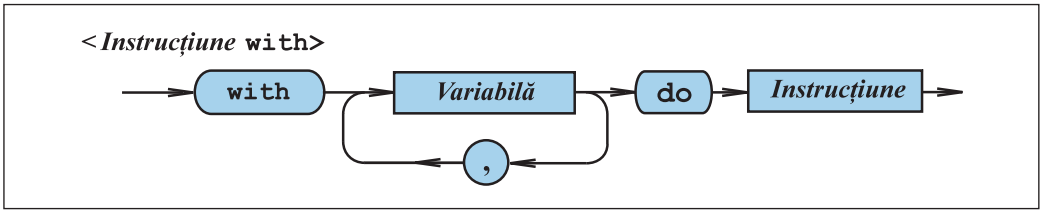


Fig. 1.6. Diagrama sintactică a instrucţiunii **`with`**

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 întîlnirea unui identificator, prima dată se testează dacă el poate fi interpretat ca un nume de cîmp al articolului respectiv. Dacă da, identificatorul va fi interpretat ca atare, chiar dacă în acel moment este accesibilă şi o variabilă avînd acelaşi 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 cîmpul `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 cîmpul `P.x` al variabilei de tip articol `P`.

Întrucît variabila de tip articol `S` nu conține niciun cîmp 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, ..., v_n **do** <*Instrucțiune*>,

unde v_1, v_2, ..., v_n sînt 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, ..., v_n trebuie specificate univoc prin denumirile cîmpurilor respective.

De exemplu, pentru variabilele `P` şi `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 obişnuit, instrucţiunea **`with`** se utilizează numai în cazurile în care se ajunge la o reducere semnificativă a textului unui program.

Întrebări şi exerciţii

❶ Indicaţi pe diagrama sintactică din *figura 1.6* drumurile care corespund instrucţiunilor **`with`** din exemplele paragrafului în studiu.

❷ Care este destinaţia instrucţiunii **`with`**?

❸ Utilizînd instrucţiunea **`with`**, excludeţi din programele `P84` şi `P85` din paragraful precedent repetările de genul

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

❹ 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 înmulţind plata pe zi cu numărul de zile lucrate. Elaboraţi un program care:

a) calculează plata pe lună a fiecărui angajat;
b) calculează salariul mediu al angajaţilor incluşi în listă;
c) afişează pe ecran datele despre angajaţii cu plata lunară maximă;
d) afişează lista angajaţilor ordonată alfabetic;
e) afişează lista angajaţilor în ordinea creşterii plăţilor pe zi;

f) ordonează lista angajaților în ordinea creşterii plăților pe lună;
g) afişează lista angajaților în ordinea creşterii numărului de zile lucrate.

❺ Un cerc poate fi definit prin coordonatele *x, y* şi raza *r*. Elaborați un program care citeşte de la tastatură datele referitoare la *n* cercuri ($n \leq 50$) şi afişează pe ecran:
a) coordonatele centrului şi raza cercului cu aria maximă;
b) numărul de cercuri incluse în cercul cu raza maximă şi coordonatele centrelor respective;
c) coordonatele centrului şi raza cercului cu aria minimă;
d) numărul de cercuri în care este inclus cercul cu raza minimă şi coordonatele centrelor respective.

1.5. Tipuri de date *mulțime* (`set`)

Un tip de date *mulțime* (**set**) se defineşte în raport cu un tip de bază care trebuie să fie ordinal:

```
<Tip mulțime> ::= [packed] set of <Tip>
```

Valorile unui tip de date **set** sînt 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 enumerîndu-i-se elementele între parantezele pătrate „[" şi „]", care țin locul acoladelor din matematică.

Notația `[]` 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, şi 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 din 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, şi 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*.

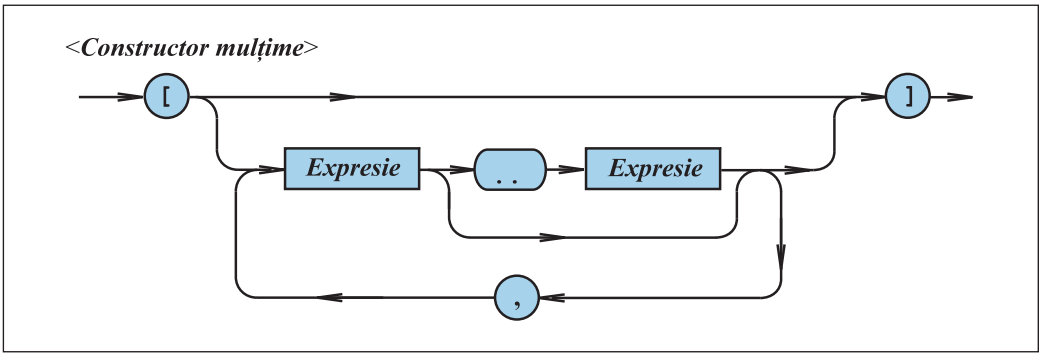


Fig. 1.7. Diagrama sintactică <*Constructor mulțime*>

Un constructor conține specificarea elementelor mulțimii, separate prin virgule şi 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 şi 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 *mulțime* se pot efectua operațiile uzuale:
+ reuniunea;
* intersecția;
- diferența,
rezultatul fiind de tip *mulțime* şi operațiile relaționale:
= egalitatea;
<> inegalitatea;

<= , >= incluziunea;
in apartenența,
rezultatul fiind de tip `boolean`.

Programul ce urmează afişează pe ecran rezultatele operațiilor +, * şi -, efectuate asupra valorilor de tip `MultimeIndicii`.

```
Program P86;
{ 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 şi articole, componentele cărora pot fi referite direct, respectiv prin indicii şi denumiri de cîmpuri, 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* măreşte viteza de execuție şi îmbunătățeşte 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 instrucţiune mai simplă:

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

Un alt exemplu sugestiv este utilizarea tipurilor de date *mulţime* în calcularea numerelor prime mai mici decît un număr natural dat *n*. Pentru aceasta se foloseşte 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 mulţimea numerelor prime;
4) din sită se elimină toţi multiplii *m* ai numărului *i*;
5) procesul se încheie cînd sita s-a golit.

```
Program P87;
{ 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.
```

Corespondenţa dintre punctele algoritmului şi instrucţiunile care le exprimă este indicată în comentariile din partea stîngă a liniilor de program.

Întrebări şi exerciţii

❶ Enumeraţi valorile posibile ale variabilelor din declaraţiile 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 Comentaţi următorul program :

```
Program P88;
{ 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 Se consideră următoarele declaraţii:

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

Care sînt valorile posibile ale variabilei NT?

❹ Scrieţi formula metalingvistică care corespunde diagramei sintactice <*Constructor mulţime*> din *figura 1.7*.

❺ Se consideră tipul de date MultimeIndicii din paragraful în studiu. Precizaţi mulţimile specificate de constructorii ce urmează:

a) `[];`

b) `[1..10];`

c) `[1..3, 9..10];`

d) `[1+1, 4..7, 9];`

e) `[3, 7..9];`

f) `[4..3];`

g) `[1..3, 7..6, 9];`

h) `[4-2..7+1];`

i) `[7-5..4+4];`

j) `[6, 9, 1..2].`

❻ Elaboraţi un program care afişează pe ecran toate submulţimile mulţimii {1, 2, 3, 4}.

❼ Elaboraţi un program care afişează pe ecran toate submulţimile mulţimii {*'A'*, *'B'*, *'C'*, *'D'*}.

❽ Se dă un şir de caractere în care cuvintele sînt separate fie prin spaţiu, fie prin caracterele *punct, virgulă, punct şi virgulă, semnul exclamării* şi *semnul întrebării*. Elaboraţi un program care afişează pe ecran cuvintele şirului de caractere citit de la tastatură.

❾ Se dă un şir de caractere. Elaboraţi un program care afişează pe ecran numărul de vocale din şir.

❿ Elaboraţi un program care citeşte de la tastatură două şiruri de caractere şi afişează pe ecran:
a) caracterele care se întîlnesc cel puţin în unul dintre şiruri;
b) caracterele care apar în ambele şiruri;
c) caracterele care apar în primul şi nu apar în şirul al doilea.

⓫ Scrieţi un program care verifică dacă numele unei persoane este introdus corect (numele este un şir de caractere ce nu conţine cifre).

⓬ În implementările actuale ale limbajului numărul valorilor tipului de bază al unui tip *mulțime* este limitat, obișnuit $n \leq 256$. În consecință, programul `P87` nu poate calcula numere prime mai mari decît 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 sînt mulțimi.

1.6. Generalități despre fișiere

Prin **fișier** se înțelege o structură de date care constă dintr-o secvență de componente. Fiecare componentă din secvență are același tip, denumit *tip de bază*. Numărul componentelor din secvență nu este fixat, însă sfîrșitul secvenței este indicat de un simbol special, notat *EOF* (*End of File* – sfîrșit de fișier). Fișierul care nu conține nicio componentă se numește *fișier vid*.

Un tip de date *fișier* se definește printr-o declarație de forma:

```
<Tip fișier> ::= [packed] file of <Tip>;
```

unde <*Tip*> este tipul de bază. Tipul de bază este un tip arbitrar, exceptînd 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 sfîrșitul secvenței, nu este o componentă a fișierului.

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, hîrtia imprimantei sau a dispozitivului de citit

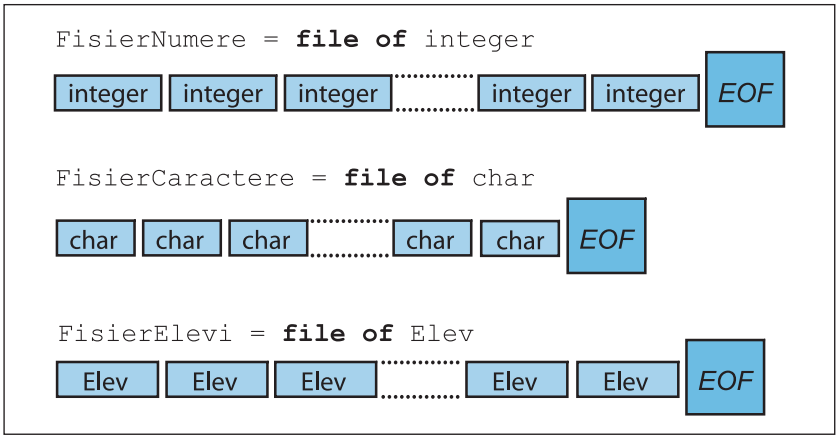


Fig. 1.8. Structura datelor de tip `FisierNume`, `FisierCaractere` şi `FisierElevi`

documente ş.a.). Informaţia pe suporturile în studiu este organizată în formă de **fişiere externe** în conformitate cu cerinţele sistemului de operare. Prin urmare, înainte de a fi utilizată, o variabilă fişier trebuie asociată cu un fişier extern. Metodele de asociere sînt specifice implementării limbajului şi sistemului de operare al calculatorului-gazdă.

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

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

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

unde *s* este o expresie de tip **`string`** care specifică calea de acces şi numele fişierului extern.

Exemple:

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

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

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

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

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

– fişierul `FE` se asociază cu un fişier extern numele căruia este citit de la tastatură şi depus în variabila de tip **`string`** `str`.

După executarea instrucţiunii `assign(f, s)` toate operaţiile referitoare la fişierul PASCAL *f* se vor efectua asupra fişierului extern *s*.

Cele mai uzuale operaţii asupra unui fişier sînt citirea şi 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 fişierului *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 fişierului *f*.

Exemple:

1) `read(FN, n);`

2) `write(FC, c);`

3) `read(FE, E).`

După **tipul operaţiilor permise** asupra componentelor, fişierele se clasifică în:
– fişiere de intrare (este permisă numai citirea);
– fişiere de ieşire (este permisă numai scrierea);
– fişiere de actualizare (sînt permise atît scrierea, cît şi citirea).

După **modul de acces** la componente, fişierele se clasifică în:
– fişiere cu acces secvenţial sau secvenţiale (accesul la componenta *i* este permis după ce s-a citit/scris componenta $i - 1$);
– fişiere cu acces aleatoriu sau direct (orice componentă se poate referi direct prin numărul ei de ordine *i* în fişier).

Menţionăm că în limbajul-standard sînt permise numai fişiere secvenţiale de intrare sau ieşire.

Tipul fişierului (de intrare, ieşire sau de actualizare) şi modul de acces (secvenţial sau direct) se stabilesc printr-o operaţie de validare, numită **deschidere a fişierului**. Pentru aceasta, în limbajul-standard se utilizează următoarele proceduri:

`reset(f)` — pregăteşte un fişier existent pentru citire;

`rewrite(f)` — creează un fişier vid şi îl pregăteşte pentru scriere.

Cînd prelucrarea componentelor se termină, fişierul trebuie închis. La **închiderea** unui fişier sistemul de operare înscrie elementul *EOF*, înregistrează fişierul extern nou-creat în directorul respectiv ş.a.m.d.

În limbajul-standard se consideră că fişierele vor fi închise implicit la terminarea execuţiei programului respectiv. În Turbo PASCAL fişierul *f* ce închide prin apelul de procedură `close(f)`.

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

1) `assign(f, s)` — asocierea fişierului PASCAL *f* cu fişierul extern *s*;
2) `reset(f)/rewrite(f)` — deschiderea fişierului *f* pentru citire/scriere;
3) `read(f, v)/write(f, e)` — citirea/scrierea unei componente a fişierului *f*;
4) `close(f)` — închiderea fişierului *f*.

După închiderea fişierului, variabila *f* poate fi asociată cu un alt fişier extern.

Întrucît valorile variabilelor de tip fişier se păstrează pe suporturile externe de informaţie, în PASCAL **atribuirile de fişiere sînt interzise**.

Întrebări şi exerciţii

❶ Explicaţi termenii *fişier PASCAL, fişier extern.*

❷ Unde se păstrează datele unui fişier PASCAL? Care este destinaţia procedurii `assign`?

❸ Reprezentaţi pe un desen 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;
```

❹ Pentru ce sînt necesare operaţiile de deschidere şi închidere a fişierelor? Cum se realizează aceste operaţii?

❺ Explicaţi destinaţia procedurilor `read` şi `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*) ?

❻ Cum se clasifică fişierele în funcţie de operaţiile permise şi modul de acces?

❼ Variabilele `A` şi `B` sînt introduse prin declaraţia

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

Este oare corectă instrucţiunea

```
A:=B
```

Argumentaţi răspunsul.

1.7. Fişiere secvenţiale

Fie definiţiile PASCAL

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

prin care au fost introduse tipul *fişier* `FT` cu tipul de bază `T`, variabila de tip *fişier* *f* şi variabila *v* de tipul `T`.

Pentru a deschide un **fişier secvenţial de ieşire**, se apelează procedura `rewrite(f)`. În continuare în fişier 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 instrucţiune de forma

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

este echivalentă cu secvenţa de instrucţiuni

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

După înscrierea ultimei componente fişierul trebuie închis.

Exemplu:

```
Program P89;
{ 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);

     { citeste 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 fişier secvenţial de intrare**, se apelează procedura `reset` (*f*) . Componenta curentă se citeşte din fişier printr-un apel de forma:

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

O instrucţiune de forma

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

este echivalentă cu secvenţa de instrucţiuni

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

Sfîrşitul de fişier este semnalizat de funcţia booleană `eof` (*f*) care ia valoarea `true` după citirea ultimei componente.

Exemplu:

```
Program P90;
{ Citirea 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;
begin
  write('Dati numele fisierului de citit:  ');
  readln(str);

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

  while not eof(FE) do
    begin
       { citeste E din fişierul FE }
       read(FE, E);
       { afiseaza E pe ecran }
       writeln(E.Nume, ' ', E.Prenume, ':',
               E.NotaMedie : 5:2);
    end;
```

```
  close(FE);          { inchide fisierul FE }
  readln;
end.
```

Subliniem faptul că numărul de componente ale unui fişier nu este cunoscut din declaraţia tipului respectiv. Teoretic, într-un fişier secvenţial de ieşire poate fi înscris un număr infinit de componente. Practic, numărul componentelor este limitat de capacitatea de memorare a suportului extern de informaţie. Citirea consecutivă a componentelor unui fişier secvenţial de intrare se încheie cînd se ajunge la elementul *EOF*.

Întrebări şi exerciţii

❶ Cîte componente poate avea un fişier? În ce ordine se scriu/citesc componentele unui fişier secvenţial?

❷ Se consideră următoarele tipuri de date:

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

Elaboraţi un program care citeşte de la tastatură datele referitoare la *n* persoane şi le înscrie într-un fişier. Creaţi fişierele `FILE1.PRS`, `FILE2.PRS`, `FILE3.PRS` care trebuie să conţină datele referitoare, respectiv, la 2, 7 şi 10 persoane.

❸ Elaboraţi un program care citeşte fişiere create de programul din exerciţiul precedent şi afişează pe ecran:
a) toate persoanele din fişier;
b) persoanele născute în anul *a*;
c) persoanele născute pe data *z.l.a*;
d) persoana cea mai în vîrstă;
e) persoana cea mai tînără.

❹ Elaboraţi un program care afişează pe ecran media aritmetică a numerelor înscrise într-un fişier de tipul **`file of`** `real`.

❺ Într-un fişier de tipul **`file of`** `char` sînt înscrise caractere arbitrare. Elaboraţi un program care afişează pe ecran numărul vocalelor din fişier.

❻ Comentaţi următorul program:

```
Program P91;
{ Eroare }
type FisierNumere = file of integer;
```

```
var FN : FisierNumere;
    i : integer;
    r : real;
    s : string;
begin
  Writeln('Dati numele fisierului de creat:  ');
  readln(s);
  assign(FN, s);
  rewrite(FN);
  i:=1;
  write(FN, i);
  i:=10;
  write(FN, i);
  r:=20;
  write(FN, r);
  close(FN);
end.
```

1.8. Fişiere *text*

E cunoscut faptul că datele fişierelor PASCAL se păstrează pe suporturile externe de informaţie. În cazul fişierelor definite prin declaraţii de forma **`file of`** *T* componentele de tip *T* sînt reprezentate pe suporturile respective în **forma internă**, şi anume, prin secvenţe de cifre binare. Acest mod de reprezentare a datelor este convenabil în cazul memoriilor externe (discurile şi benzile magnetice, discurile optice ş.a.). În cazul echipamentelor de intrare–ieşire (tastatura, ecranul, imprimanta etc.) datele respective trebuie reprezentate în **forma externă**, şi anume, prin secvenţe de caractere.

Pentru a facilita interacţiunea între om şi sistemul de calcul, în PASCAL informaţia destinată utilizatorului se reprezintă în formă de fişiere *text*. Un fişier *text* este format dintr-o secvenţă de caractere divizată în linii (*fig. 1.9*). Lungimea liniilor este variabilă. Sfîrşitul fiecărei linii este indicat de un element special, notat *EOL* (*End Of Line* – sfîrşit de linie). Întrucît lungimea liniilor este

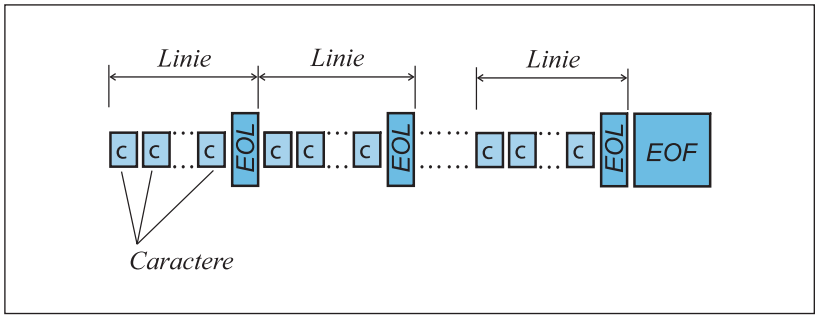


Fig. 1.9. Structura unui fişier *text*

variabilă, poziția unei linii în cadrul fișierului nu poate fi calculată din timp. În consecință, accesul la componentele fișierelor *text* este **secvențial**.

Un fișier *text* se definește printr-o declarație de forma:

```
var f : text;
```

tipul predefinit `text` fiind cunoscut oricărui program PASCAL. Subliniem faptul că tipurile `text` şi **`file of`** `char` sînt distincte, întrucît fișierul **`file of`** `char` nu include elemente *EOL* (*fig. 1.8*).

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

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

`readln(`*f*`)` – trece la linia următoare.

Sfîrşitul de linie este semnalat de funcția booleană `eoln(`*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 şi extragerea datelor, de regulă, se utilizează fișierele *text* predefinite `Input` şi `Output`. Fișierul `Input` este destinat numai pentru operații de citire şi este asociat cu fișierul de intrare al sistemului de operare (de regulă, tastatura). Fișierul `Output` este destinat numai pentru operații de scriere şi este asociat cu fișierul-standard de ieşire al sistemului de operare (de regulă ecranul). Aceste fișiere sînt deschise şi închise automat la începutul şi, respectiv, la sfîrşitul 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 programul `P93` care creează pe discul curent fișierul text `FILE.TXT`. Liniile fișierului sînt introduse de la tastatură (fișierul `Input`). Sfîrşitul liniei se indică prin acționarea tastei <*ENTER*>, iar sfîrşitul fișierului prin acționarea tastelor <*CTRL+Z* >, <*ENTER*>.

```
Program P93;
{ Crearea fisierului text FILE.TXT }
var F : text;
    c : char;
begin
  assign(F, 'FILE.TXT'); { asociaza F cu FILE.TXT }
  rewrite(F);          { deschide fisierul F pentru scriere }
  while not eof do       { eof(Input) }
    begin
      while not eoln do  { eoln(Input) }
        begin
          read(c);       { citeste c din Input }
          write(F, c);   { scrie c in F }
        end;
      writeln(F);        { scrie EOL in F }
      readln;          { trece la linia urmatoare din Input }
    end;
  close(F);              { inchide F }
end.
```

Programul ce urmează afişează conţinutul fişierului `FILE.TXT` pe ecran.

```
Program P94;
{ Citirea fisierului text FILE.TXT }
 var F : text;
     c : char;
begin
  assign(F, 'FILE.TXT'); { asociaza F cu FILE.TXT }
  reset(F);              { deschide F pentru citire }
  while not eof(F) do
    begin
      while not eoln(F) do
        begin
          read(F, c);    { citeste c din F }
          write(c);      { scrie c in Output }
        end;
      readln(F);         { trece la linia urmatoare din F }
      writeln;           { scrie EOL in Output }
    end;
  close(F);              { inchide F }
  readln;
end.
```

Explorarea caracter cu caracter a fişierelor *text* este greoaie în situaţia cînd secvenţele de caractere din text trebuie interpretate ca formînd date de tip `integer`, `real`, `boolean`, *şir de caractere*. Conversiunea între forma externă

şi reprezentarea internă a acestor tipuri ar cădea în sarcina programatorului. În consecinţă, procedurile de citire/înscriere sînt extinse în felul următor.

În cazul fişierelor *text* variabila *v* dintr-un apel `read(f, v)` poate fi de tipul `integer`, `real`, `char` sau *şir de caractere*. La citire secvenţa 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 *şir de caractere*. La scriere valoarea respectivă este transformată din reprezentarea internă într-o secvenţă de caractere.

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

Pentru exemplificare prezentăm programul `P95` care citeşte de la tastatură cîte trei numere reale *a, b, c* pe care le scrie în fişierul `IN.TXT`. Apoi citind aceste trei numere, reprezentînd laturile unui triunghi, scrie în fişierul `OUT.TXT` numerele *a, b* şi *c*, semiperimetrul *p* şi aria triunghiului *s*. În continuare, conţinutul fişierului `OUT.TXT` este afişat pe ecran.

```
Program P95;
{ Prelucrarea fisierelor IN.TXT si OUT.TXT }
var F, G : text;
    a, b, c, p, s : real;
    str : string;
begin
  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 }

  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 in G }
    end;

  close(F);                { inchide F }
  close(G);                { inchide G }

  reset(G);                { deschide G pentru citire }
  while not eof(G) do
    begin
      readln(G, str);      { citeste str din G }
      writeln(str);        { afiseaza 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 `P95` afişează pe ecran:

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

Întrebări şi exerciţii

❶ Care este diferenţa dintre un fişier *text* şi un fişier **`file of`** `char`?

❷ Explicaţi semnificaţia elementelor *EOL* şi *EOF*.

❸ Care este diferenţa dintre procedurile `read` şi `readln`? Dar dintre procedurile `write` şi `writeln`?

❹ Lansaţi în execuţie următorul program:

```
Program P96;
{ Asocierea fisierului FN cu consola }
type FisierNumere = file of integer;
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.
```

Explicați rezultatele afișate pe ecran.

❺ Elaborați un program care afișează pe ecran conținutul oricărui fișier *text*.

❻ Elaborați un program care afișează pe ecran numărul de vocale dintr-un fișier *text*.

❼ Datele de intrare ale unui program sînt 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.

❽ Datele de intrare ale unui program sînt 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.

❾ Fiecare linie a unui fișier *text* conține următoarele date, separate prin spații:
– numărul de ordine (`integer`);
– numele (un **`string`** ce nu conține spații);
– prenumele (un **`string`** ce nu conține spații);
– nota la disciplina 1 (`real`);
– nota la disciplina 2 (`real`);
– nota la disciplina 3 (`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 (`integer`);
– numele (**`string`**);
– prenumele (**`string`**);
– nota medie (`real`).

Fișierul creat în punctul *c* trebuie afișat pe ecran.

Test de autoevaluare nr. 1

1. Se consideră declarațiile

```
type Obiect = (      Istoria, Geografia, Matematica,
                Informatica, Fizica);
     Nota = 1..10;
     SituatiaScolara = array [Obiect] of Nota;
```

Reprezentați pe un desen structura datelor de tipul `SituatiaScolara`.

2. Precizați tipul indicilor şi tipul componentelor tipului de dată `OrarulLectiilor` din declarațiile ce urmează:

```
type ZiDeScoala = (L, Ma, Mi, J, V, S);
     Lectie = 1..6;
     Obiect = (LimbaRomana, LimbaModerna, Istoria,
               Geografia, Matematica,Informatica, Fizica,
               Chimia);
     OrarulLectiilor = array [ZiDeScoala, Lectie] of Obiect;
```

3. Se consideră declarațiile:

```
type Tablou = array [1..10] of integer;
var x, y : Tablou;
```

Scrieți expresia aritmetică a cărei valoare este:
a) suma primelor patru componente ale variabilei `x`;
b) suma ultimelor patru componente ale variabilei `y`;
c) valoarea absolută a componentei a treia a variabilei `x`;
d) valoarea absolută a componentei a şasea a variabilei `y`;
e) suma primei componente a variabilei `x` şi a ultimei componente a variabilei `y`.

4. Se consideră n ($n \leq 50$) numere întregi a_1, a_2, a_3, ..., a_n. Elaborați un program PASCAL care citeşte numerele respective de la tastatură şi le afişează pe ecran în ordinea inversă citirii: a_n, ..., a_3, a_2, a_1.

5. Ce operații pot fi efectuate asupra şirurilor de caractere de tip `string`? Precizați tipul rezultatelor acestor operații.

6. Elaborați un program PASCAL care afişează pe ecran în ordine inversă şirul de caractere citit de la tastatură. De exemplu, şirul `'soare'` va fi afişat la ecran ca `'eraos'`.

7. Reprezentați pe un desen structura datelor de tipul `Data` şi `Persoana` din declarațiile ce urmează:

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

8. Se consideră următoarele tipuri de date:

```
type Angajat = record
                 NumePrenume : string;
                 Salariu : real;
               end;
     ListaDePlata = array[1..100] of Angajat;
```

Elaborați un program care citeşte de la tastatură datele despre *n* ($n \leq 100$) angajați şi afişează pe ecran informațiile despre angajatul (angajații) care au cel mai mare salariu.

9. Care este destinația instrucțiunii **with**?

10. Enumerați valorile posibile ale variabilelor din declarațiile ce urmează:

```
var V : set of 'X'..'Z';
    I : set of 8..9;
```

11. Se consideră şiruri de caractere formate din literele mari ale alfabetului latin. Elaborați un program care afişează pe ecran numărul de vocale din şirul de caractere *S* citit de la tastatură.

12. Unde se păstrează datele fişierelor PASCAL? Care este destinația procedurii `assign`?

13. Cum se clasifică fişierele în funcție de operațiile permise şi modul de acces?

14. Se consideră următoarele tipuri de date:

```
type Angajat = record
                 NumePrenume : string;
                 Salariu : real;
               end;
     FisierAngajati = file of Angajat;
```

Elaborați un program care creează fişierul `SALARII.DAT` şi înscrie în acest fişier datele referitoare la *n* angajați. Datele respective se citesc de la tastatură.

15. Elaborați un program PASCAL care afişează pe ecran conținutul fişierului `SALARII.DAT` creat în itemul precedent.

16. Datele de intrare ale unui program sînt înmagazinate în fişierul *text* `REZULTAT.TXT`. Fiecare linie a fişierului conține cîte două numere reale separate prin spațiu şi unul din cuvintele `BUN`, `DEFECT`. Elaborați un program care:

a) creează fişierul *text* `MEDIA.TXT` liniile căruia conțin media celor două numere reale şi cuvintele `BUN` sau `DEFECT` din liniile respective ale fişierului de intrare;

b) afişează pe ecran liniile fişierului creat.

Capitolul 2

INFORMAȚIA

2.1. Cantitatea de informație

Sensul uzual al cuvîntului **informație** „ştire, comunicare verbală, scrisă sau transmisă prin alte metode despre anumite fapte, evenimente, activităţi 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 $\{s_1, s_2, ..., s_n\}$. Se consideră că valorile curente ale variabilei S nu sînt cunoscute din timp. E cunoscută numai mulțimea $\{s_1, s_2, ..., s_n\}$, **denumită mulțimea mesajelor posibile**.

De exemplu, semaforul de circulație poate fi reprezentat ca o sursă de informație, mulțimea de mesaje posibile ale căruia este {*verde, galben, roşu*}. Aparatul de telegrafiat reprezintă o sursă de informație, mulțimea de mesaje posibile ale căruia include literele *A, B, C, ..., Z,* cifrele *0, 1, 2, ..., 9* şi semnele de punctuație. Mesajele posibile ale tastaturii sînt: „*Este acționată tasta A*”, „*Este acționată tasta B*”, ..., „*Este acționată tasta F1*”, „*Este acționată tasta F2*”, ..., „*Sînt acționate concomitent tastele CTRL şi 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.

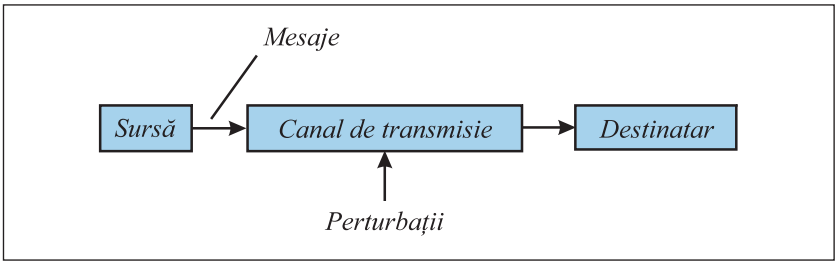


Fig. 2.1. Schema generală a unui sistem de transmisie a informației

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

$$\log_a 2 = 1 \quad (bit)$$

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

$$I = \log_2 n \ (bit).$$

În *tabelul 2.1* sînt 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 cîteva exemple. Cantitatea de informație a unui mesaj de semafor este de

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

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

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

Cantitatea de informație a unei litere a alfabetului grec {*A, B, Γ, Δ , ..., Ω*}, $n = 24$, este de

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

Dacă se cunoaşte 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 multiplii unui bit:

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

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

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

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

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

Întrebări şi exerciții

❶ Cum se defineşte o sursă de informație? Dați cîteva 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 şi ce semnificație are ea?

❺ Determinați cantitatea de informație într-un mesaj al surselor cu următoarele mesaje posibile:
a) literele mari şi mici ale alfabetului latin;
b) literele mari şi mici ale alfabetului grec;
c) literele mari şi mici ale alfabetului român;
d) cifrele zecimale 0, 1, 2, ..., 9;
e) cifrele 0, 1, 2, ..., 9, semnele +, –, ×, / şi 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* – secundele) 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 şi decodificarea informaţiei

Se numeşte **semn** un element al unei mulţimi finite de obiecte ce se pot distinge. O mulţime de semne ordonate liniar se numeşte **alfabet**.

Prezentăm în continuare unele dintre nenumăratele alfabete folosite de oameni:

a) alfabetul cifrelor zecimale: 0, 1, 2, ..., 9;

b) alfabetul literelor latine mari: *A, B, C, ..., Z*;

c) mulţimea semnelor zodiacului;

d) mulţimea fazelor lunii.

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

Prezentăm cîteva exemple de alfabete binare:

a) cifrele {*0, 1*};

b) perechea de culori {*roşu, galben*};

c) perechea de stări {*închis, deschis*};

d) perechea de răspunsuri {*da, nu*};

e) perechea de tensiuni {0*V*, 2*V*};

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 şir finit din m semne, dintre care unele se pot repeta, formează un **cuvînt**, m reprezentînd **lungimea cuvîntului**. 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*-poziţionale**. În continuare prezentăm unele mulţimi de cuvinte cu lungime constantă:

1-poziţionale: {0, 1};

2-poziţionale: {00, 01, 10, 11};

3-poziţionale: {000, 001, 010, 011, 100, 101, 110, 111};

4-poziţionale: {0000, 0001, ..., 1110, 1111}.

Se observă că cuvintele $(m + 1)$-poziţionale se formează cîte două din cuvintele m-poziţionale prin adăugarea cifrelor binare 0 şi 1. Prin urmare, mulţimea cuvintelor m-poziţionale include 2^m cuvinte distincte.

Cuvintele binare se utilizează pentru reprezentarea, transmiterea, păstrarea şi prelucrarea mesajelor $s_1, s_2, ..., s_n$ ale sursei de informaţie (*fig. 2.2*).

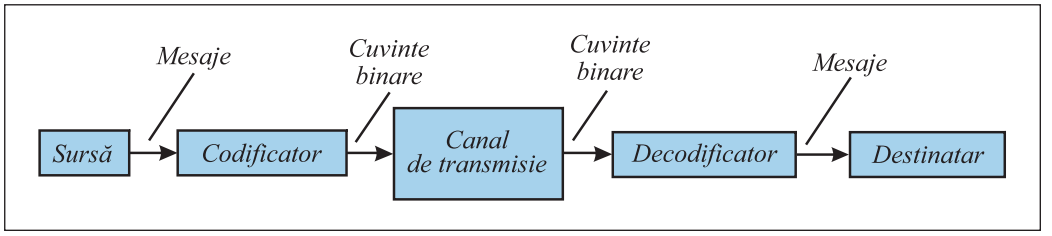


Fig. 2.2. Codificarea şi decodificarea mesajelor în sistemele de transmisie a informaţiei

Regula de transformare a mesajelor în cuvinte se numeşte cod, iar operaţia respectivă – codificare. Operaţia inversă codificării se numeşte decodificare. Dispozitivele tehnice care realizează operaţiile în cauză se numesc, respectiv, codificator şi decodificator.

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

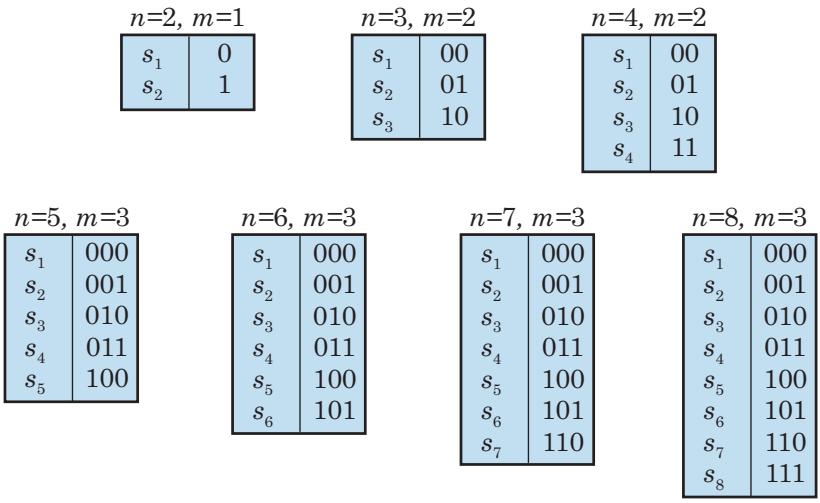


Fig. 2.3. Coduri de cuvinte cu lungime constantă (coduri *m*-poziţionale)

Operaţiile de codificare şi decodificare constau în extragerea datelor necesare din tabel. Evident, decodificarea va fi univocă numai atunci cînd cuvintele binare incluse în tabel sînt distincte. Acest lucru este posibil dacă lungimea m a cuvintelor de cod satisface inegalitatea

$$\mathbf{2}^{\boldsymbol{m}} \geq \boldsymbol{n}.$$

După logaritmare obţinem:

$$\boldsymbol{m} \geq \mathbf{log}_2\, \boldsymbol{n}.$$

Întrucît expresia $\log_2 n$ exprimă cantitatea de informaţie, se poate afirma:

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

De exemplu, lungimea cuvintelor pentru codificarea literelor mari ale alfabetului latin {*A, B, C, ...Z*}, n = 26, se determină din relaţia

$$\boldsymbol{m} \geq \mathbf{log}_2\, \mathbf{26} \approx \mathbf{4{,}700}.$$

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

A – 00000
B – 00001
C – 00010

D – 00011
E – 00100
...
Z – 11001.

Un astfel de cod a fost propus de filosoful şi omul de stat englez Francis Bacon încă în anul 1580.

Algoritmii de elaborare a **codurilor cu cuvinte de lungime variabilă** sînt mult mai complicaţi şi se studiază în cursurile avansate de informatică.

Întrebări şi exerciţii

❶ Ce este un alfabet? Daţi exemple de alfabete binare.
❷ Cum se reprezintă semnele oricărui alfabet binar?
❸ Explicaţi cum pot fi formate cuvintele binare (m+1)-poziţionale. Care este numărul cuvintelor binare m-poziţionale distincte?
❹ Care este destinaţia unui cod? Cum se defineşte codul m-poziţional?
❺ Cum se efectuează codificarea şi decodificarea mesajelor în cazurile cînd codul este definit printr-un tabel?
❻ Codificaţi mesajele s_3, s_4 şi s_6 ale unei surse cu 7 mesaje posibile. Utilizaţi codul 3-poziţional din *figura 2.3*.
❼ Decodificaţi mesajele 100, 000 şi 010, reprezentate în codul 3-poziţional 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 Utilizînd codul 3-poziţional din *figura 2.3*, n = 6, codificaţi şirul de mesaje s_1, s_2, s_6, s_5, s_3, s_6, s_3, s_2, s_1.
❿ Cum influenţează cantitatea de informaţie a unui mesaj asupra lungimii cuvintelor de cod?
⓫ Explicaţi sensul termenilor **cantitate de informaţie** şi **informaţie**.
⓬ Elaboraţi un program care codifică şi decodifică literele alfabetului latin. Se va utiliza codul propus de Francis Bacon.
⓭ Elaboraţi un program care alcătuieşte tabelul unui cod m-poziţional pentru o sursă cu n mesaje posibile.

2.3. Coduri frecvent utilizate

Orice cod, utilizat pentru reprezentarea, transmiterea, păstrarea şi prelucrarea informaţiei, trebuie să fie econom şi insensibil la perturbaţii, iar echipamentele respective de codificare şi decodificare – să fie simple. Pe parcursul dezvoltării tehnicii de calcul au fost elaborate mai multe coduri. Aceste coduri se clasifică în coduri numerice şi coduri alfanumerice.

Codurile numerice oferă posibilitatea reprezentării cifrelor {0, 1, 2, ..., 9} prin cuvinte binare 4-poziţionale. Exemple de coduri numerice sînt prezentate în *tabelul 2.2*.

Tabelul 2.2

Coduri numerice

Cifra	*Denumirea codului*			
	Direct	*Gray*	*Aiken*	*Exces 3*
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 şi 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	*Cuvînt binar*	*Echivalent zecimal*	*Simbol*	*Cuvînt binar*	*Echivalent zecimal*
Spaţiu	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
/	0101111	47	_	1011111	95

Simbol	*Cuvînt binar*	*Echivalent zecimal*	*Simbol*	*Cuvînt binar*	*Echivalent zecimal*
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	1100101	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-poziţional şi include $2^7 = 128$ de simboluri. Primele 32 de simboluri (cuvintele binare 0000000, 0000001, 0000010, …, 0011111) specifică detaliile tehnice ale transmisiunilor de informaţii şi nu au fost incluse în tabel. Cuvintele binare 0100000, 0100001, 0100010, …, 1111110 reprezintă caracterele impri-

mabile din textele în limba engleză. Cuvîntul 1111111 reprezintă caracterul neimprimabil *Delete* (Anulare).

Codificarea mesajelor se realizează prin înlocuirea simbolurilor cu cuvintele binare respective. De exemplu, cuvîntul 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 cuvîntul STOP.

De regulă, limbajele de programare operează nu cu cuvintele binare propriuzise, 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'.`

Orientat la textele engleze, codul ASCII nu include literele cu semne diacritice şi caracterele grafice speciale întîlnite în diferite limbi europene şi în lucrările ştiințifice. De aceea pentru calculatoarele moderne s-au elaborat versiuni dedicate ale codului ASCII, denumite **coduri ASCII extinse**. Codurile extinse sînt 8-poziționale şi includ $2^8 = 256$ de simboluri. Structura codurilor respective este prezentată în *tabelul 2.4*.

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 şi include simbolurile de la 128 la 255. Aceste simboluri sînt utilizate pentru reprezentarea literelor alfabetelor naționale, precum şi a caracterelor ştiințifice frecvent utilizate. Pentru exemplificare, în *tabelul 2.4* sînt prezentate codurile literelor *Ă, ă, Â, â, Î, î, Ş, ş, Ţ, ţ* din alfabetul limbii române propuse în anul 1992 de firma *TISH* (Chişinău). 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 şi a multiplilor lui pentru măsurarea cantității de informație:

1 *octet* = $2^3 = 8$ *biți*;	1 *Gigaoctet* = $2^{30} \approx 10^9$ *octeți*;
1 *Kilooctet* = $2^{10} \approx 10^3$ *octeți*;	1 *Teraoctet* = $2^{40} \approx 10^{12}$ *octeți*;
1 *Megaoctet* = $2^{20} \approx 10^6$ *octeți*;	1 *Petaoctet* = $2^{50} \approx 10^{15}$ *octeți*.

În literatura de specialitate octetul este notat prin *B* (*byte*), iar multiplii respectivi prin *KB, MB, GB, TB şi PB*.

Tabelul 2.4

Structura codurilor ASCII extinse

Simbol	*Cuvînt binar*	*Echivalent zecimal*	*Observaţii*
Spaţiu	00100000	32	Partea 1: – simbolurile codului ASCII
!	00100001	33	
"	00100010	34	
#	00100011	35	
…	…	…	
}	01111101	125	
~	01111110	126	
Del	01111111	127	
А	10000000	128	Partea a 2-a: – simboluri specifice limbilor naţionale; – caractere grafice; – caractere ştiinţifice
Б	10000001	129	
В	10000010	130	
…	…	…	
/	11110000	240	
Ă	11110001	241	
ă	11110010	242	
Â	11110011	243	
â	11110100	244	
Î	11110101	245	
î	11110110	246	
Ş	11110111	247	
ş	11111000	248	
Ţ	11111001	249	
¯	11111010	250	
Ö	11111011	251	
Ţ	11111100	252	
ţ	11111101	253	
□	11111110	254	
	11111111	255	

Întrebări şi exerciţii

❶ Cîte mesaje posibile pot fi codificate cu ajutorul unui cod m-poziţional?

❷ Determinaţi codurile admise de sistemul de operare cu care lucraţi dvs.

❸ Codificaţi în codul *Gray* următoarele şiruri de cifre zecimale: 123, 461, 952, 783, 472.

❹ Decodificaţi mesajele reprezentate în codul *Aiken*:

a) `0011` `1111` `0100` *b)* `1110` `0010` `1101`

c) `1111` `0000` `0100`

d) `0010` `0001` `1011`

e) `0011` `1100` `1111`

f) `1111` `1101` `0000`

❺ Codificaţi în codul ASCII expresiile:

a) `A+B`

b) `FOR I=1 TO N`

c) `PRINT A$`

d) `NEXT I`

e) `PAUSE`

f) `PROGRAM`

❻ Decodificaţi mesajele 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 Elaboraţi un program care afişează 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 operaţiilor aritmetice;
e) caracterele speciale `;`, `<`, `=`, `>`, `?`, `[`, `]`, `{`, `}`, `/`, `\`.

2.4. Informaţia mesajelor continue

Sursele de informaţie studiate pînă în prezent se definesc printr-o variabilă S, care poate lua valori dintr-o mulţime finită de elemente distincte $\{s_1, s_2, ..., s_n\}$, denumită mulţimea mesajelor posibile. Practica ne demonstrează că nu toate sursele de informaţie 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 ieşire ale microfonului etc.) care poate lua orice valoare într-un anumit interval $[s_{min}, s_{max}]$.

Sursele de informaţie care sînt definite printr-o variabilă S ce ia valori dintr-o mulţime 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 şi în cazul surselor discrete, mesajele continue se produc în timp. Prin urmare, S este o funcţie de timp, $S = S(t)$. În scopul evaluării cantităţii de informaţie în mesajele continue, vom analiza valorile funcţiei $S(t)$ numai în momentele de timp $t_1, t_2, ..., t_m$ (*fig. 2.4*). Valorile date, notate prin $S(t_1), S(t_2), ..., S(t_m)$, se numesc **eşantioane** (din franceză *echantillion* „cantitate mică luată dintr-un produs pentru a determina calitatea lui”).

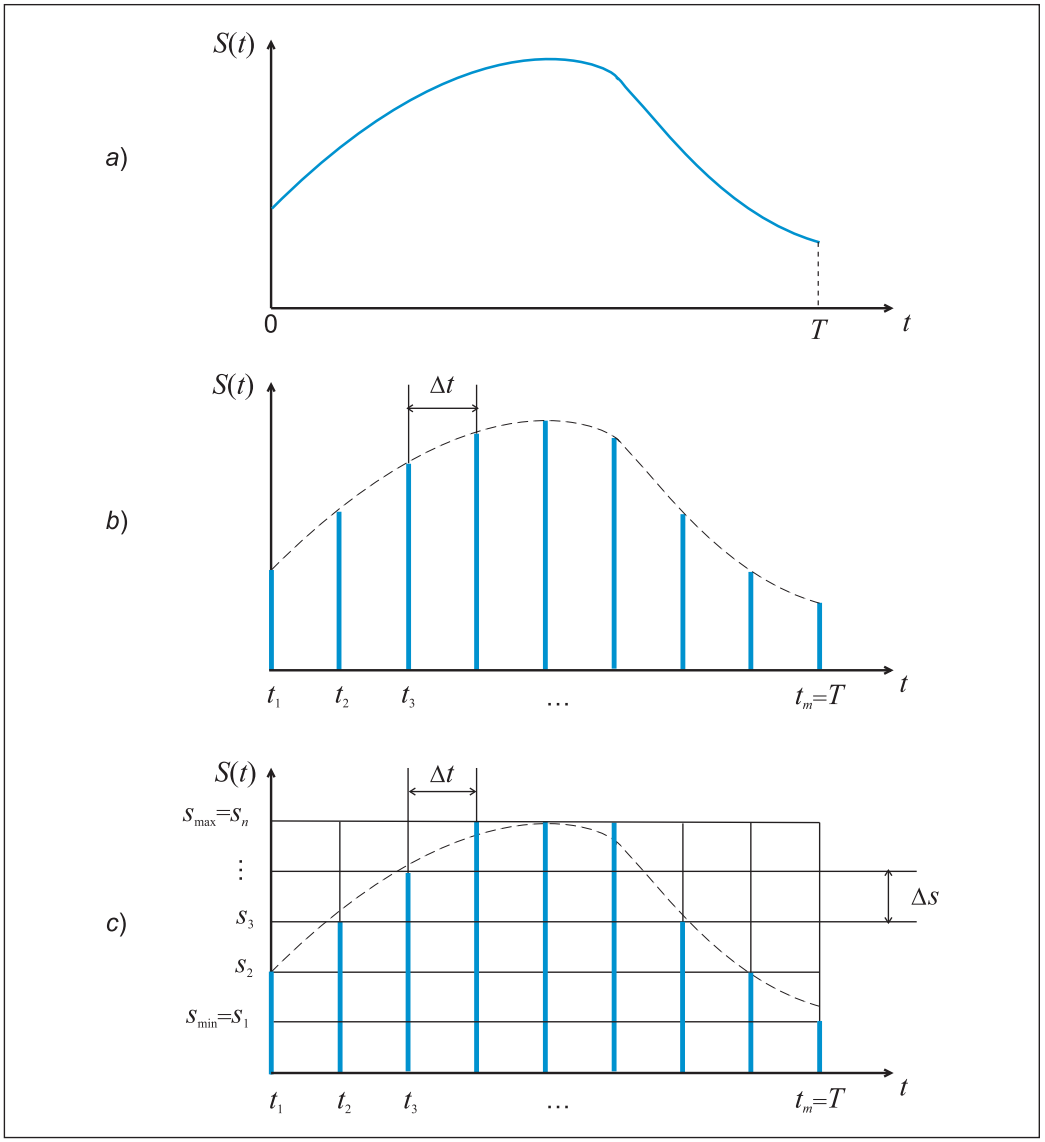


Fig. 2.4. Discretizarea mesajelor continue: *a* – mesaj continuu; *b* – mesaj discretizat în timp (eşantionat); *c* – mesaj eşantionat şi discretizat în valoare (cuantificat)

De obicei, momentele de timp t_1, t_2, ..., t_m se definesc conform relației:

$$t_i = t_{i-1} + \Delta t.$$

Mărimea Δt se numeşte **perioada de eşantionare**. Valoarea concretă a perioadei de eşantionare 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 şi $\Delta t = 1\ h$, iar în tehnica prelucrării semnalelor sonore $\Delta t = 5 \cdot 10^{-5}\ s$.

Operația de transformare a mesajelor continue în eşantioane se numeşte discretizare în timp sau eşantionare.

Evident, pînă la recepţia mesajului continuu, valorile concrete ale eşantioanelor $S(t_1)$, $S(t_2)$, ..., $S(t_m)$ sînt necunoscute destinatarului. Se cunoaşte numai intervalul $[s_{min}, s_{max}]$ în care poate lua valori arbitrare orice eşantion. În scopul evaluării cantităţii de informaţie dintr-un eşantion, vom rotunji valoarea $S(t_i)$ la una din valorile prestabilite s_1, s_2, ..., s_n (*fig. 2.4*).

Valorile prestabilite s_1, s_2, ..., s_n se numesc **cuante**, iar operaţia de transformare a valorilor curente ale mesajelor continue în cuante se numeşte **discretizare în valoare** sau **cuantificare**.

De obicei,

$$s_1 = s_{min}, \quad s_2 = s_{min} + \Delta s, \quad \ldots, \quad s_i = s_{i-1} + \Delta s, \quad \ldots, \quad s_n = s_{max}.$$

Mărimea Δs reprezintă **pasul** sau **intervalul de cuantificare**. Valoarea concretă a intervalului de cuantificare depinde de natura fizică a sursei de informaţie, precizia de măsurare, puterea de rezoluţie a destinatarului etc.

De exemplu, pentru un termometru medical $s_{min} = 34^o$, $s_{max} = 42^o$ şi $\Delta s = 0{,}1^o$. În meteorologie $s_{min} = -60^o$, $s_{max} = +60^o$, $\Delta s = 1^o$. Pentru vitezometrul unui automobil $s_{min} = 0$, $s_{max} = 150$ km/h, $\Delta s = 5$ km/h.

Numărul eşantioanelor m şi numărul cuantelor n se determină din următoarele relaţii (*fig. 2.4 b* şi *c*):

$$m = \frac{T}{\Delta t} + 1; \quad n = \frac{|s_{max} - s_{min}|}{\Delta s} + 1,$$

unde T este durata mesajului continuu.

Întrucît cuantele s_1, s_2, ..., s_n pot fi considerate mesaje discrete, cantitatea de informaţie într-un eşantion

$$I = \log_2 n = \log_2 \left(\frac{|s_{max} - s_{min}|}{\Delta s} + 1 \right),$$

iar cantitatea de informaţie 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

$$I = \log_2 \left(\frac{|42 - 34|}{0{,}1} + 1 \right) \approx 6{,}34 \text{ biţi},$$

iar pentru vitezometrul automobilului

$$I = \log_2 \left(\frac{|150 - 0|}{5} + 1 \right) \approx 4{,}95 \text{ biţi}.$$

Cantitatea de informaţie dintr-o înregistrare audio, pentru care $n = 256$, $\Delta t = 5 \cdot 10^{-5}$ s şi $T = 45$ min. este

$$V = \frac{45 \cdot 60}{5 \cdot 10^{-5}} \log_2 256 = 4{,}32 \cdot 10^8 \text{ biţi} = 432 \text{ Mbiţi}.$$

Dacă precizia de măsurare şi puterea de rezoluţie a destinatarului cresc, perioada de eşantionare Δt şi pasul de cuantificare Δs pot fi micşorate. În consecinţă, va creşte şi cantitatea de informaţie dintr-un mesaj continuu.

Informaţia mesajelor continue poate fi reprezentată printr-un set de cuvinte binare. Pentru aceasta, cuantele s_1, s_2, ..., s_n se codifică exact la fel ca şi oricare alte mesaje discrete. De exemplu, indicaţiile unui termometru medical ($I \approx 6{,}34$ *biţi*) pot fi codificate cu un cod 7-poziţional. Cel mai frecvent se utilizează codurile numerice directe (*tab. 2.2*), cuvîntul de cod reprezentînd numărul cuantei respective. În unele aplicaţii se utilizează codul *Gray*, care este insensibil la perturbaţii.

Dispozitivul care transformă mesajul continuu aplicat la intrare într-o succesiune de cuvinte de cod se numeşte **convertor analog-numeric**. Operaţia inversă, şi 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 informaţia de procesat este reprezentată prin mesaje continue: controlul proceselor tehnologice, dirijarea obiectelor aflate în mişcare, monitorizarea parametrilor fiziologici în medicină, filtrarea şi mixajul semnalelor audio etc.

Întrebări şi exerciţii

❶ Care este diferenţa dintre sursele cu mesaje discrete şi sursele cu mesaje continue?

❷ Daţi cîteva exemple de surse cu mesaje continue. Concretizaţi intervalul în care ia valori variabila ce defineşte sursa.

❸ Explicaţi cum se efectuează operaţia de eşantionare. Cum se alege perioada de eşantionare?

❹ Explicaţi cum se efectuează operaţia de cuantificare. Cum se alege pasul de cuantificare?

❺ Cum influenţează valorile perioadei de eşantionare şi ale pasului de cuantificare cantitatea de informaţie extrasă dintr-un mesaj continuu?

❻ Altimetrul cu impulsuri al unui avion poate măsura înălţimi de la 100 m pînă la 20 km. Eroarea de măsurare nu depăşeşte 1 m. Pentru a efectua o măsurare sînt necesare 10^{-3} *s*. Determinaţi cantitatea de informaţie furnizată de altimetru timp de 5 ore de zbor.

❼ Temperatura din interiorul unui reactor chimic se înregistrează pe o bandă de hîrtie milimetrică. Pe axa absciselor se indică timpul (1 mm reprezintă o oră), iar pe axa ordonatelor – temperatura (1 mm reprezintă 10°C). Cîtă informaţie conţine o înregistrare efectuată timp de 30 de zile, dacă temperatura poate varia de la 80 pînă la 1000°C ?

❽ Pentru înregistrarea sunetului se utilizează microfoane, tensiunea de ieşire a cărora variază de la 0 pînă la 100 μV. Aparatul de înregistrare nu distinge tensiunile valorile cărora diferă cu mai puţin de 0,1 μV. Pentru a asigura o reproducere fidelă, în fiecare secundă se iau 40 000 de eşantioane. Cîtă informaţie va furniza microfonul dat timp de 3 ore?

❾ Care este destinaţia convertoarelor analog-numerice şi numeric-analogice?

❿ Elaboraţi un program care introduce de la tastatură valorile curente ale eşantioanelor şi afişează pe ecran codurile cuantelor corespunzătoare.

⓫ Elaboraţi un program care simulează funcţionarea unui convertor numeric-analog.

2.5. Cuantizarea imaginilor

Imagine se numeşte reprezentarea unui obiect, executată pe o suprafaţă prin acţiunea 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 informaţie, imaginea este împărţită în **microzone**, numite de cele mai multe ori **puncte** sau **pixeli**. Descompunerea imaginii în puncte se realizează cu ajutorul unui **rastru** (de la cuvîntul latin *raster* „greblă”). Rastrul reprezintă o suprafaţă plană, în general dreptunghiulară, pe care sînt trasate două seturi de linii paralele, perpendiculare între ele (*fig. 2.5*). Densitatea liniilor şi, respectiv, densitatea punctelor caracterizează puterea de rezoluţie a echipamentelor pentru reproducerea sau formarea imaginilor.

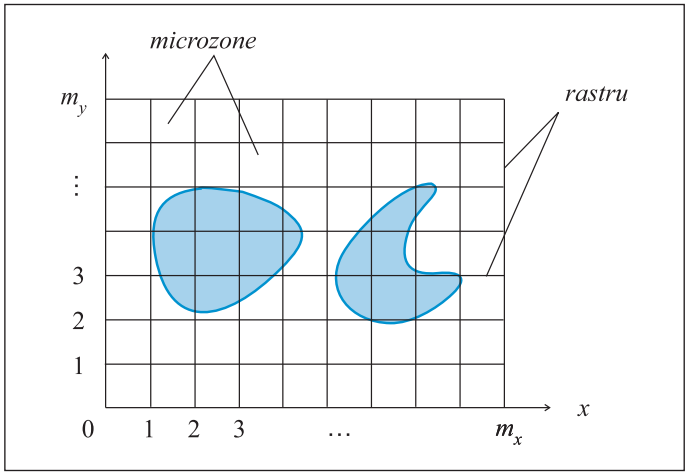


Fig. 2.5. Descompunerea imaginii în microzone

De exemplu, pentru ilustraţiile de gazetă se folosesc rastre cu rezoluţia 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, $800 \times \times 600$, 720×400, ..., 1024×1024 de puncte.

Descompunerea imaginii în puncte (microzone) reprezintă o procedură de discretizare în spaţiu.

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*).

Întrucît culorile pot fi redate 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 durata 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:

$$I = 3 \cdot 625 \cdot 625 \cdot \log_2 32 \approx 5{,}6\ \textit{Mbiți}.$$

Un film color cu durata de 1,5 ore va conține:

$$V = 1{,}5 \cdot 3\,600 \cdot 25 \cdot I \approx 791\ \textit{Gbiți}.$$

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-numerice. Imaginile de pe hîrtie pot fi cuantizate cu ajutorul unui dispozitiv special, numit **scaner**. Acest dispozitiv conține celule fotosensibile, convertoare analog-numerice și mecanisme de avansare a hîrtiei.

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 baleere în vizualizatoare, matricea de ace în imprimantele mecanice etc.

Întrebări și exerciții

❶ Numiți operațiile necesare pentru a cuantiza imaginea.

❷ Care este destinația rastrului? Din care considerente se alege densitatea liniilor unui rastru?

❸ Cum se evaluează cantitatea de informație dintr-o imagine monocromă?

❹ Cum pot fi redate culorile unei imagini multicolore? Cum se evaluează cantitatea de informație dintr-o imagine color?

❺ 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.

❻ Cîtă informație conține o fotografie color cu dimensiunile 20×20 cm, reprodusă cu ajutorul unui rastru ce conține 60 de puncte/cm? Pot fi redate pînă la 256 de niveluri de luminanță ale punctelor respective.

❼ Rastrul unei camere de luat vederi este format din $1\,024 \times 1\,024$ de puncte. Pot fi redate pînă la 64 de niveluri de luminanță ale punctelor respective. Cîtă informație va conține un film video cu durata de 3 ore?

❽ Imaginea numerică conține cîte un cuvînt binar pentru fiecare punct al rastrului unui vizualizator. Cîte niveluri de luminanță pot fi redate pe ecran dacă cuvintele imaginii numerice sînt 3-poziționale? 5-poziționale? 8-poziționale?

2.6. Reprezentarea şi transmiterea informației

Obiectul material folosit pentru păstrarea, transmiterea sau prelucrarea informației se numeşte **purtător de informație**. Deosebim purtători statici şi 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 hîrtia. Mesajele înregistrate pe hîrtie sub formă de manuscrise, desene sau texte tipărite pot fi păstrate un timp foarte îndelungat. În calculatoare, ca purtători statici, se utilizează:

– hîrtia 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 foloseşte:

– unde acustice în gaze (aer) sau lichide;
– tensiuni şi curenți electrici;
– unde electromagnetice etc.

Întrucît transmiterea informației se produce în spațiu şi timp, cel puțin o mărime fizică a purtătorului de informație trebuie să se schimbe.

> ***Se numeşte semnal variația mărimii fizice ce asigură transmiterea mesajelor. Caracteristica semnalului folosită pentru reprezentarea (descrierea) mesajelor se numeşte parametru informațional al semnalului.***

De exemplu, în transmisiunile radio şi TV, în calitate de purtători se utilizează undele electromagnetice. Amplitudinea sau frecvența acestor unde poate varia în timp (*fig. 2.6*). 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 foloseşte, de obicei, curentul electric, tensiunea şi intensitatea curentului fiind para-

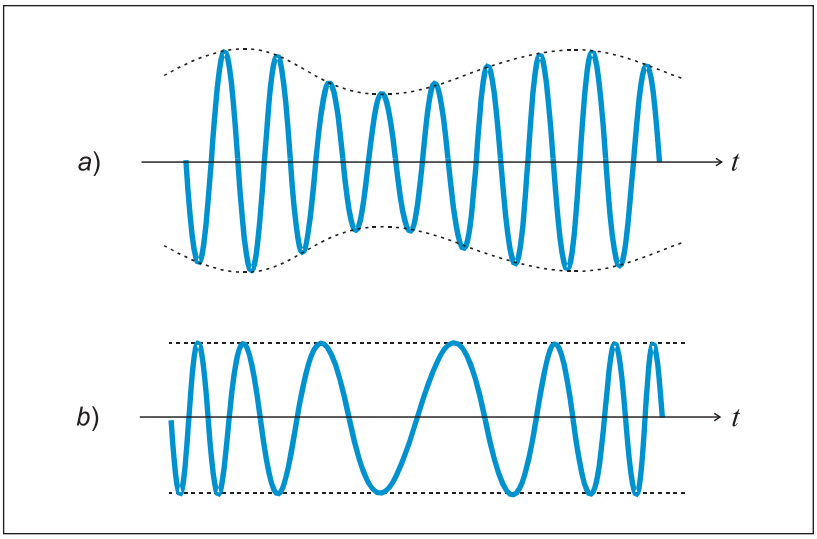


Fig. 2.6. Parametrii informaționali ai undelor electromagnetice:
a – amplitudinea; *b* – frecvența

metrii 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 şi 1 pot fi de asemenea asociate, respectiv, cu absență sau prezență de impuls sau cu impuls negativ şi impuls pozitiv.

Semnalul se numeşte discret, dacă parametrul informaţional respectiv poate lua numai un număr finit de valori. Semnalul se numeşte continuu dacă parametrul informaţional poate lua orice valoare într-un anumit interval.

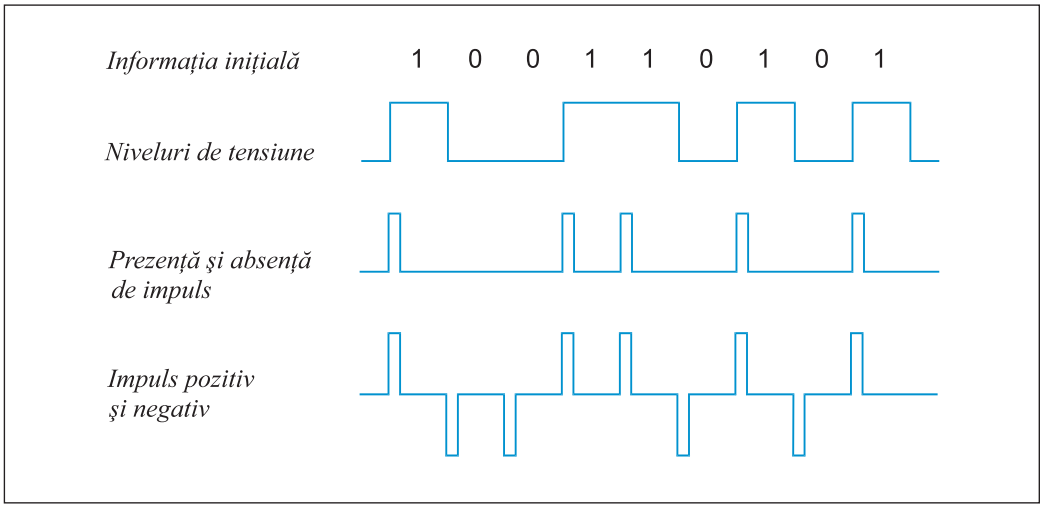


Fig. 2.7. Semnale utilizate în tehnica de calcul

De exemplu, semnalele din *figura 2.6* sînt semnale continue, iar semnalele din *figura 2.7* sînt semnale discrete. Evident, semnalele discrete şi continue sînt forme de reprezentare a mesajelor respective.

Orice sistem tehnic utilizează acele semnale care-i asigură o realizare cît mai bună a funcţiilor pentru care a fost conceput. Calculatoarele actuale utilizează niveluri de tensiune, reţelele telefonice – curenţi electrici, iar radioul şi televiziunea – unde electromagnetice etc. Practica demonstrează că semnalele continue pot fi transmise la distanţe mult mai mari decît cele discrete. Prin urmare, pentru a asigura legătura între calculatoarele aflate la distanţă, la emisie semnalele discrete vor fi transformate în semnale continue. La recepţie se va realiza transformarea inversă: semnalul continuu va fi transformat într-un semnal discret.

Operaţia prin care parametrul informaţional al semnalului continuu se modifică în funcţie de valorile semnalului discret se numeşte modulare.

Dispozitivul tehnic care realizează operaţia în cauză se numeşte **modulator**. Pentru exemplificare, în *figura 2.8* sînt prezentate operaţiile de modulare în amplitudine şi frecvenţă.

Operaţia de extragere a semnalului discret dintr-un semnal continuu în funcţie de procedeul de modulare adoptat se numeşte demodulare.

Dispozitivul tehnic care realizează operaţia respectivă se numeşte **demodulator**.

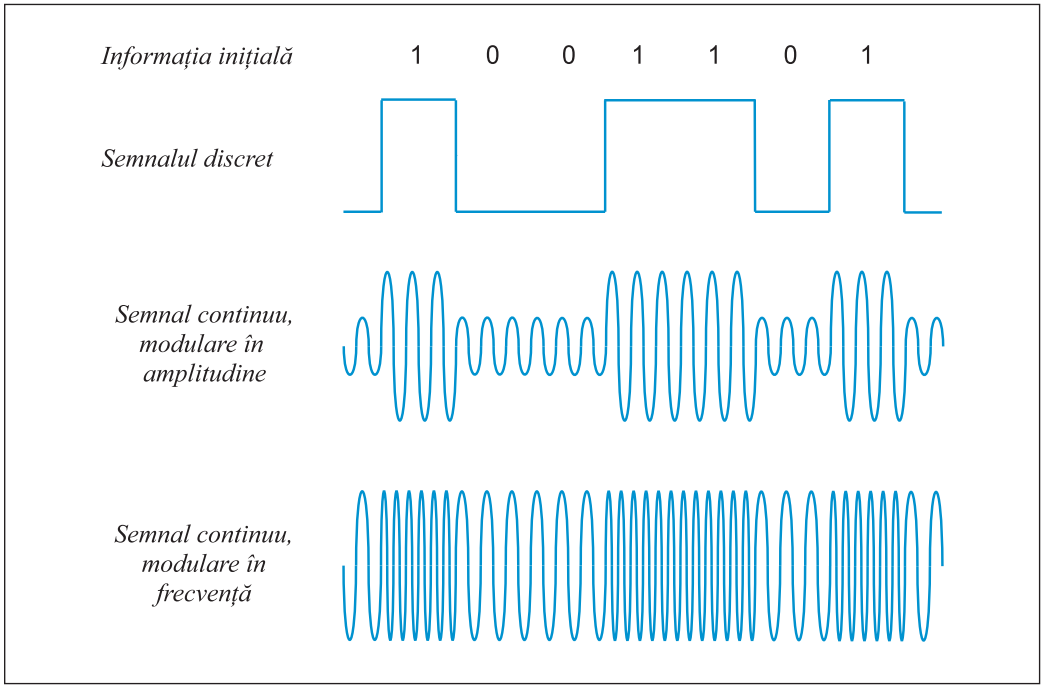


Fig. 2.8. Modularea semnalelor continue

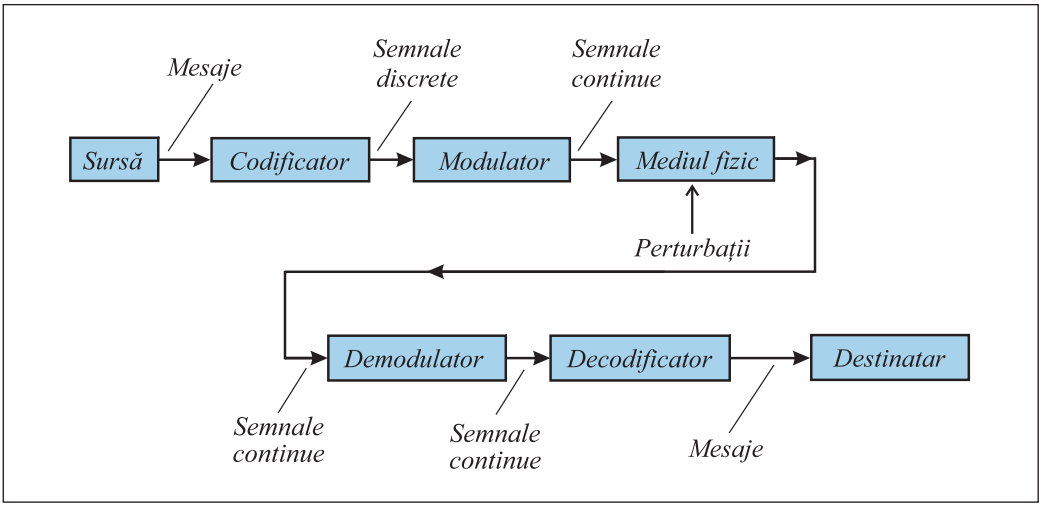


Fig. 2.9. Schema detaliată a unui sistem de transmisie a informației

Schema detaliată a sistemului de transmisie a informației este prezentată în *figura 2.9*.

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 şi 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 – circa 1 *Gbit*/s; capacitatea unui canal optic – 1 *Tbit*/s.

Întrebări şi exerciții

❶ Care este deosebirea dintre purtătorii statici şi purtătorii dinamici de informație?

❷ Determinați tipul următorilor purtători de informație:
a) cartele perforate;
b) unde ultrasonore;
c) benzi perforate;
d) unde acustice;
e) pelicule fotosensibile;

f) unde gravitaţionale;
g) hîrtie fotografică.

❸ Numiţi parametrii informaţionali ai semnalelor emise de următoarele surse:
a) microfonul;
b) staţia de radio, unde lungi, medii sau scurte;
c) instrumentul muzical;
d) staţia de radio, unde ultrascurte;
e) camera de luat vederi.

❹ Descrieţi purtătorii de informaţie şi semnalele utilizate în calculatoarele actuale.

❺ Explicaţi operaţiile de modulare şi demodulare a semnalelor.

❻ Care este destinaţia modulatorului? Dar a demodulatorului?

❼ În *figura 2.8* sînt redate semnalele continue ce reprezintă cuvîntul binar 1001101. În codul ASCII (*tab. 2.3*) acestui cuvînt îi corespunde simbolul `M`. Redaţi pe un desen semnalele continue ce corespund următoarelor simboluri:

a) `>`; *e)* `<`;
b) `r`; *f)* `K`;
c) `W`; *g)* `a`;
d) `9`; *h)* `@`.

❽ De ce depinde capacitatea de transmisie a canalului? În ce unităţi se măsoară ea?

❾ Cum influenţează perturbaţiile asupra capacităţii de transmisie a canalului?

Test de autoevaluare nr. 2

1. Scrieţi mulţimea cuvintelor 3-poziţionale formate din cifrele binare {0, 1}.

2. În codul propus de filosoful englez Francis Bacon literele mari ale alfabetului latin se reprezintă astfel:

A – 00000, B – 00001, C – 00010, D – 00011, ..., Z – 11001.

Decodificaţi, utilizînd *codul Bacon*, cuvintele binare 00010, 00000, 00010, 00001. Codificaţi, utilizînd acelaşi cod, cuvîntul BAC.

3. Se consideră o sursă de informaţie, mulţimea de mesaje posibile ale căreia este formată din literele mari şi mici ale alfabetului latin $S = \{A, B, C, ..., Z, a, b, c, ..., z\}$. Determinaţi lungimea minimă m a cuvintelor binare ce vor asigura codificarea şi decodificarea univocă a mesajelor.

4. Determinaţi lungimea minimă m a cuvintelor binare ce vor asigura codificarea şi decodificarea univocă a mesajelor numerice ale unui calendar electronic. Mesajele respective au forma *zz.ll.aa*, unde *zz* reprezintă ziua, *ll* – luna, iar *aa* – anul (ultimele două cifre).

5. Cîtă informaţie se conţine într-un simbol al codului ASCII extins?

6. Considerînd că textul va fi reprezentat în codul ASCII extins, determinaţi cantitatea de informaţie într-o dictare scrisă timp de 10 min. de un elev, care este capabil să scrie circa 200 de caractere pe minut.

7. Elaborați un program PASCAL care afișează pe ecran codurile caracterelor citite de la tastatură.

8. Elaborați un program PASCAL care afișează pe ecran caracterele ce corespund codurilor citite de la tastatură.

9. Calculați cîtă informație conține o fotografie color cu dimensiunile 10×10 cm, reprodusă cu ajutorul unui rastru ce conține 30 de puncte/cm. Pot fi redate pînă la 128 de niveluri de luminanță ale punctelor respective.

10. Selectați din lista ce urmează purtătorii statici de informație:

a) cartela perforată;
b) unda ultrasonoră;
c) banda magnetică;
d) pelicula fotosensibilă;
e) unda electromagnetică;
f) cartela magnetică;
g) curentul electric;
h) hîrtia pentru imprimantă.

11*. Termometrul medical asigură măsurarea temperaturilor din intervalul 34–42° cu o precizie de 0,1°. Pe parcursul unei zile temperatura pacientului a fost măsurată de 3 ori. Cîtă informație conțin rezultatele acestor măsurări?

12*. Calculați cîtă informație conține o înregistrare audio cu perioada de eşantionare $\Delta t = 3 \cdot 10^{-5}$ s, durata de $T = 30$ min. şi numărul de cuante $n = 128$.

13*. Indicați corespondența între noțiunile din coloana stîngă şi definițiile din coloana dreaptă:

(1) cod;
(2) semnal;
(3) informație;
(4) alfabet;

(a) o ştire, o comunicare verbală sau scrisă;
(b) o mulțime de numere binare;
(c) o mulțime ordonată de semne;
(d) un anumit număr de biți;
(e) regula de transformare a mesajelor în cuvinte binare;
(f) variația mărimii fizice ce asigură transmiterea mesajelor.

14*. Indicați corespondența între denumirile componentelor (coloana din stînga) ale unui sistem de transmisie a informației şi destinația acestora (coloana din dreapta):

(1) codificatorul;
(2) modulatorul;
(3) demodulatorul;
(4) decodificatorul;

(a) transformă semnale continue în semnale discrete;
(b) transformă secvențele sonore în cuvinte binare;
(c) transformă mesajele emise de sursă în cuvinte binare;
(d) transformă imaginile grafice în cuvinte binare;
(e) transformă cuvintele binare în mesaje;
(f) transformă semnalele discrete în semnale continue.

* Numai pentru profilul real.

Capitolul 3

BAZELE ARITMETICE ALE TEHNICII DE CALCUL

3.1. Sisteme de numerație

În calculatoarele digitale informația de orice categorie este reprezentată, stocată şi 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 defineşte baza sistemului de numerație.***

Prezentăm cîteva 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 şi 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 şi 2;

– **sistemul octal** este un sistem de numerație în baza 8, conținînd 8 cifre: 0, 1, 2, ..., 7;

– **sistemul hexazecimal** este un sistem de numerație în baza 16 şi conține 16 cifre: 0, 1, 2, ..., 9, *A* (zece), *B* (unsprezece), *C* (doisprezece), *D* (treisprezece), *E* (paisprezece), *F* (cincisprezece).

În *tabelul 3.1* sînt date reprezentările unora şi aceloraşi 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:

$$(3856{,}43)_{10} = 3\cdot10^3 + 8\cdot10^2 + 5\cdot10^1 + 6\cdot10^0 + 4\cdot10^{-1} + 3\cdot10^{-2}.$$

Se observă că în această reprezentare semnificația (valoarea) fiecărei cifre depinde de poziția pe care o ocupă în număr. De exemplu, cifra 3 se întîlneşte de 2 ori: prima dată are semnificația 3000, 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:

$$N = c_n c_{n-1} \ldots c_1 c_0, c_{-1} c_{-2} \ldots c_{-m}.$$

Valoarea acestui număr se evaluează în funcție de baza sistemului:

$$(N)_b = c_n b^n + c_{n-1} b^{n-1} + \ldots + c_1 b^1 + c_0 b^0 + c_{-1} b^{-1} + c_{-2} b^{-2} + \ldots + c_{-m} b^{-m}.$$

Efectuînd calculele respective, se va realiza **conversiunea** numărului $(N)_b$ din baza b în sistemul zecimal.

De exemplu,

$(101{,}1)_{10} = 1\cdot10^2 + 0\cdot10^1 + 1\cdot10^0 + 1\cdot10^{-1} = 101{,}1;$

$(101{,}1)_2 = 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-1} = 5{,}5;$

$(101{,}1)_3 = 1\cdot3^2 + 0\cdot3^1 + 1\cdot3^0 + 1\cdot3^{-1} = 10{,}333\ldots;$

$(101{,}1)_8 = 1\cdot8^2 + 0\cdot8^1 + 1\cdot8^0 + 1\cdot8^{-1} = 65{,}125;$

$(101{,}1)_{16} = 1\cdot16^2 + 0\cdot16^1 + 1\cdot16^0 + 1\cdot16^{-1} = 257{,}0625.$

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 mîinilor.

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 şi logice;

– materializarea fizică a cifrelor în procesul prelucrării sau stocării numerelor se face mai uşor pentru două simboluri decît pentru zece: perforat–neperforat, contact închis–contact deschis, prezență sau absență de curent etc.;

– circuitele care trebuie să diferențieze numai între două stări sînt mai sigure în funcționare decît cele care trebuie să diferențieze între zece stări.

Menționăm că în procesul dezvoltării civilizației umane au fost create şi **sisteme de numerație nepoziționale**. Drept exemplu poate servi **sistemul roman**, care utilizează cifrele *I, V, X, L, C, D, M*. Întrucît regulile de reprezentare a numerelor şi de efectuare a operațiilor aritmetice sînt foarte complicate, sistemele nepoziționale au o utilizare foarte restrînsă.

Întrebări şi exerciții

❶ Cum se defineşte un sistem de numerație?

❷ Care este deosebirea dintre sistemele de numerație poziționale şi nepoziționale?

❸ Dați exemple de sisteme de numerație poziționale. Cum se defineşte baza sistemului de numerație?

❹ Evaluați numărul $(101,1)_b$ scris în următoarele sisteme de numerație:

$b = 4, 5, 6, 7, 9, 11, 12, 13, 14$ şi 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 PASCAL care evaluează numerele scrise în baza b, $b \le 10$.

❽ Elaborați un program care evaluează numerele scrise în baza b, $10 \le b \le 36$.

3.2. Conversiunea numerelor dintr-un sistem în altul

Conversiunea numărului $(N)_b$ în echivalentul său zecimal se efectuează conform formulei din paragraful precedent:

$$(N)_b = c_n b^n + c_{n-1} b^{n-1} + \ldots + c_1 b^1 + c_0 b^0 + c_{-1} b^{-1} + c_{-2} b^{-2} + \ldots + c_{-m} b^{-m}.$$

Conversiunea numărului zecimal $(N)_{10}$ în echivalentul său în baza b se efectuează conform următoarelor reguli:

– se împarte la baza respectivă partea întreagă şi cîturile obţinute după fiecare împărţire, pînă se obţine cîtul zero; rezultatul conversiunii părţii întregi este constituit din resturile obţinute, considerate în ordinea inversă de apariţie;

– se înmulţeşte cu baza partea fracţionară, apoi toate părţile fracţionare obţinute din produsul anterior, pînă cînd partea fracţionară a unui produs este zero sau pînă la obţinerea unui număr de cifre fracţionare dorit; rezultatul conversiunii părţii fracţionare este constituit din părţile întregi ale produselor, considerate în ordinea apariţiei.

Să analizăm cîteva exemple.

1) Să se transforme numărul zecimal 53,40625 în echivalentul său binar.

$53 : 2 = 26 + {}^1/_2$;
$26 : 2 = 13 + {}^0/_2$;
$13 : 2 = 6 + {}^1/_2$;
$6 : 2 = 3 + {}^0/_2$;
$3 : 2 = 1 + {}^1/_2$;
$1 : 2 = 0 + {}^1/_2$.

Prin urmare, partea întreagă a numărului binar va fi 110101.

$0{,}40625 \times 2 = 0{,}8125$;
$0{,}8125 \times 2 = 1{,}625$;
$0{,}625 \times 2 = 1{,}25$;
$0{,}25 \times 2 = 0{,}5$;
$0{,}5 \times 2 = 1{,}0$.

Partea fracţionară 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.

$23 : 2 = 11 + {}^1/_2$;
$11 : 2 = 5 + {}^1/_2$;
$5 : 2 = 2 + {}^1/_2$;
$2 : 2 = 1 + {}^0/_2$;
$1 : 2 = 0 + {}^1/_2$;

$0{,}7 \times 2 = 1{,}4$;
$0{,}4 \times 2 = 0{,}8$;

$0{,}8 \times 2 = 1{,}6;$
$0{,}6 \times 2 = 1{,}2;$
$0{,}2 \times 2 = 0{,}4;$
$0{,}4 \times 2 = 0{,}8;$
...

Se observă că operația poate fi continuată la infinit, adică nu există o conversiune exactă a numărului analizat. Prin urmare,

$$(23{,}7)_{10} = (10111{,}101100...)_2.$$

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 şi exerciții

❶ Cum se efectuează conversiunea unui număr dintr-un sistem cu baza b în sistemul zecimal?

❷ Transformați în sistemul zecimal numerele ce urmează:

a) $(100001{,}01111)_2$;
b) $(328{,}678)_9$;
c) $(100{,}100)_{16}$;
d) $(10{,}01)_{16}$;
e) $(AAA{,}BBB)_{16}$;
f) $(EE{,}00F)_{16}$;
g) $(1221{,}1112)_3$;
h) $(1321{,}1312)_4$;
i) $(124{,}521)_7$;
j) $(4231{,}124)_5$;
k) $(50{,}505050)_6$;
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 efectuînd 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 efectuînd 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, efectuînd 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 şi invers

Întrucît $8 = 2^3$, **conversiunea binar-octală** şi **octal-binară** se poate face direct. Orice cifră octală se reprezintă prin 3 cifre binare:

0 = 000;	4 = 100;
1 = 001;	5 = 101;
2 = 010;	6 = 110;
3 = 011;	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 cîte 3 cifre binare pornind de la poziția virgulei spre stînga pentru partea întreagă, respectiv dreapta pentru partea fracționară, aflînd 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 şi 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 şi exerciţii

❶ Cum se efectuează conversiunea octal-binară şi binar-octală?

❷ Transformaţi 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 Transformaţi 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 Elaboraţi un program pentru conversiunea binar-octală şi octal-binară.

❺ Cum se efectuează conversiunea hexazecimal-binară şi binar-hexazecimală?

❻ Transformaţi în sistemul binar numerele hexazecimale ce urmează:

a)	FFF,AAA;	*e)*	3,1AB;	*i)*	C,DC1;
b)	F,1A;	*f)*	AABB,0000F;	*j)*	942,14A;
c)	1,009;	*g)*	81,91A;	*k)*	CAA,B;
d)	0,00F;	*h)*	10,01;	*l)*	DAD,ABA.

❼ Transformaţi 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 Elaboraţi un program pentru conversiunea binar-hexazecimală şi hexazecimal-binară.

❾ Transformaţi 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.

❿ 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.

⓫ Elaborați un program pentru conversiunea octal-hexazecimală şi hexazecimal-octală.

3.4. Operații aritmetice în binar

Operațiile aritmetice cu numerele binare sînt foarte simple. Regulile de operare în sistemul binar sînt prezentate în *tabelele 3.2, 3.3* şi *3.4.*

Exemple:

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 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1

Exemple:

1) Se propune adunarea în binar a numerelor zecimale 29 şi 43:

$(29)_{10} = (11101)_2$;

$(43)_{10} = (101011)_2$;

$$\begin{array}{r} 11101 \\ +\ 101011 \\ \hline 1001000 \end{array}$$

Verificare: $(1001000)_2 = (72)_{10}$, rezultatul este corect, întrucît $(29)_{10} + (43)_{10} = (72)_{10}$.

2) Se propune scăderea în binar a numărului zecimal 37 din numărul zecimal 46:

$(37)_{10} = (100101)_2$;

$(46)_{10} = (101110)_2$;

$$\begin{array}{r} 101110 \\ -\ 100101 \\ \hline 1001 \end{array}$$

Verificare: $(1001)_2 = (9)_{10}$, rezultatul este corect, întrucît $(46)_{10} - (37)_{10} = (9)_{10}$.

3) Se propune înmulțirea în binar a numerelor zecimale 3,25 şi 7,125:
$(3,25)_{10} = (11,01)_2$;
$(7,125)_{10} = (111,001)_2$;

$$\begin{array}{r} 11,01 \\ 111,001 \\ \hline 1101 \\ 0000 \\ 0000 \\ 1101 \\ 1101 \\ 1101 \\ \hline 10111,00101 \end{array} \times$$

Verificare: $(10111,00101)_2 = (23,15625)_{10}$, rezultatul este corect, întrucît $(3,25)_{10} \times$ $\times (7,125)_{10} = (23,15625)_{10}$.

4) Se propune împărțirea în binar a numărului zecimal 211 la numărul zecimal 3:
$(211)_{10} = (11010011)_2$;
$(3)_{10} = (11)_2$;

$$\begin{array}{r|l} 11010011 & 11 \\ \cline{2-2} 11 & 1000110 \\ \cline{1-1} 00100 & \\ 11 & \\ \cline{1-1} 11 & \\ 11 & \\ \cline{1-1} 01 & \end{array}$$

Deci 11010011 : 11=1000110, rest 1.
Verificare: $(1000110)_2 = (70)_{10}$, rezultatul este corect, întrucît $(211)_{10} : (3)_{10} =$ $= (70)_{10} + (1)_{10}$.
Menționăm că atît la înmulțire, cît şi la împărțire, fixarea virgulei care desparte partea întreagă de cea fracționară se face la fel ca şi în sistemul de numerație zecimal.

Întrebări şi exerciții

❶ Cum se execută operațiile aritmetice în sistemul binar?
❷ Calculați în sistemul binar:

a)	34 + 251;	*d)*	14 × 8;	*g)*	2015 + 1995;
b)	68 – 7;	*e)*	63 : 3;	*h)*	28,5 + 0,75;
c)	1512 + 620;	*f)*	1996 – 51;	*i)*	63,125 – 4,125;

j)	$3{,}0625 \times 2{,}125$;	*n)*	$32 : 2$;	*r)*	401×4;
k)	$0{,}5 \times 0{,}5$;	*o)*	$32 : 16$;	*s)*	$32 : 4$;
l)	$1 : 0{,}5$;	*p)*	401×8;	*t)*	401×2;
m)	$40 : 0{,}125$;	*q)*	$32 : 8$;	*u)*	$933 : 3$.

Numerele analizate sînt scrise în sistemul zecimal.

❸ Elaboraţi un program pentru adunarea şi scăderea numerelor binare.

❹ Elaboraţi un program pentru înmulţirea şi împărţirea numerelor binare.

3.5. Reprezentarea numerelor naturale în calculator

Calculatoarele actuale utilizează sistemul de numeraţie binar. Reprezentarea numerelor naturale $N = \{0, 1, 2, ...\}$ se realizează pe un număr fix de poziţii binare, de regulă, 8, 16, 32 sau 64 (*fig. 3.1*).

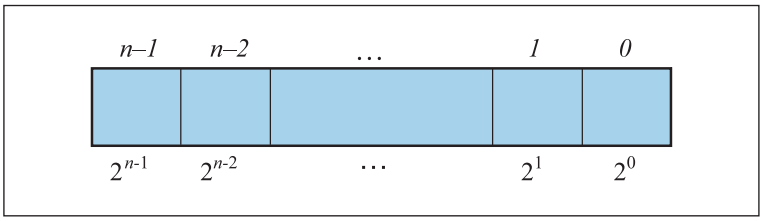


Fig. 3.1. Reprezentarea numerelor naturale pe *n* poziţii binare

În poziţiile 0, 1, ..., *n*–1 sînt înscrise cifrele binare ale numărului natural reprezentat în sistemul de numeraţie binar. Alinierea numerelor binare se realizează la dreapta, eventualele zerouri nesemnificative sînt plasate în faţa numărului binar.

Drept exemplu, în *figura 3.2* este redată reprezentarea numărului natural

$$1039 = (10000001111)_2$$

pe un număr de 16 poziţii binare.

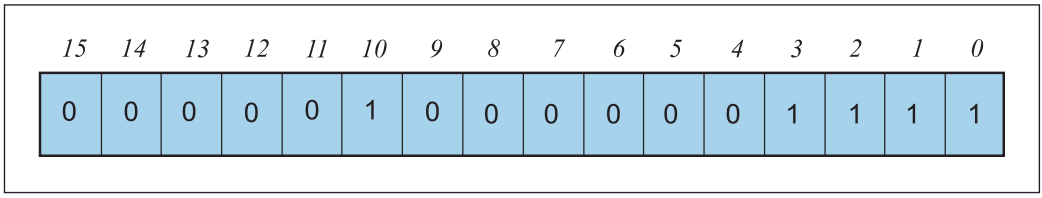


Fig. 3.2. Reprezentarea numărului natural 1039 pe 16 poziţii binare

Numărul maxim ce poate fi reprezentat pe *n* poziţii binare (*fig. 3.3*) este

$$1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + ... + 1 \cdot 2^{n-1} = 2^n - 1.$$

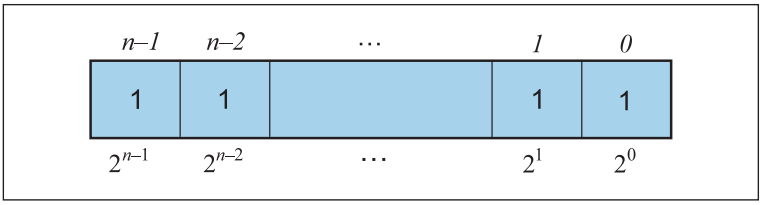


Fig. 3.3. Reprezentarea numărului natural maxim pe *n* poziții binare

Prin urmare, pe *n* poziții binare pot fi reprezentate numere naturale din intervalul $[0; 2^n-1]$.

Întrebări şi 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, 1024, 2048, 4096, 65 535 pe 16 poziții binare.
❹ Calculați numerele naturale maxime ce pot fi reprezentate pe 4, 8, 12, 16, 24, 32 şi 64 de poziții binare.

3.6. Reprezentarea numerelor întregi

În calculator nu există posibilitatea introducerii directe a semnelor + şi –, ataşate numerelor pozitive şi negative. Din acest motiv, reprezentarea semnului numărului *x* se face cu ajutorul unei cifre binare, denumită **cifră-semn**, aşezată în poziția *n*–1 (*fig. 3.4*):

$$S = \begin{cases} 0, & x > 0; \\ 1, & x < 0. \end{cases}$$

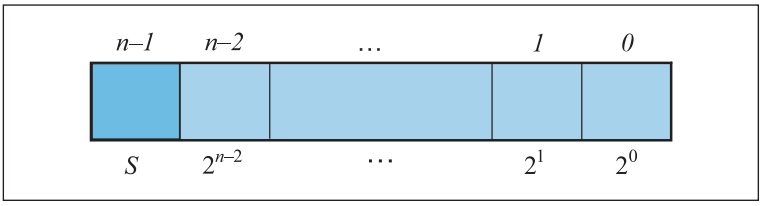


Fig. 3.4. Reprezentarea numerelor întregi pe *n* poziții binare

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 şi semn). Scrierea unui număr în acest cod este foarte simplă: în cifra-semn se scrie 0, dacă numărul este pozitiv, şi 1, dacă el este negativ; în partea de valoare se înscrie numărul în sistemul binar obişnuit.

Drept exemplu, în *figura 3.5* sînt redate reprezentările numerelor +52 şi –52 pe 8 poziţii binare.

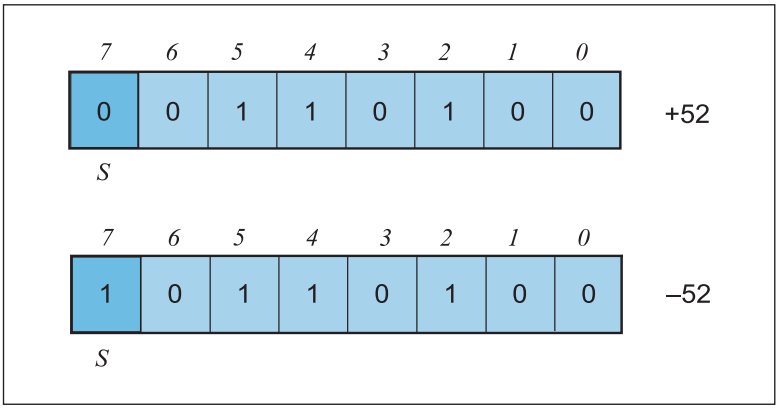


Fig. 3.5. Reprezentarea numerelor +52 şi –52 în cod direct

În codul direct, pe n poziţii binare se pot reprezenta numere întregi pozitive şi negative N, astfel încît:

$$-(2^{n-1}-1) \le N \le (2^{n-1}-1).$$

Menţionăm că în codul direct există două reprezentări binare pentru numărul 0: 00...0 şi 10...0. Codul direct este rar utilizat în calculatoare, deoarece necesită algoritmi complicaţi de executare a operaţiilor aritmetice şi 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 întîi ca şi cum ar fi pozitiv, apoi se inversează fiecare cifră binară, adică 1 devine 0 şi 0 devine 1.

În *figura 3.6* sînt redate reprezentările numerelor +52 şi –52 în cod invers pe 8 poziţii binare.

Se poate uşor constata că intervalul numerelor întregi N care se pot reprezenta în cod invers este acelaşi ca şi pentru reprezentarea în cod direct, iar numărul 0 are două reprezentări binare: 00...0 şi 11...1.

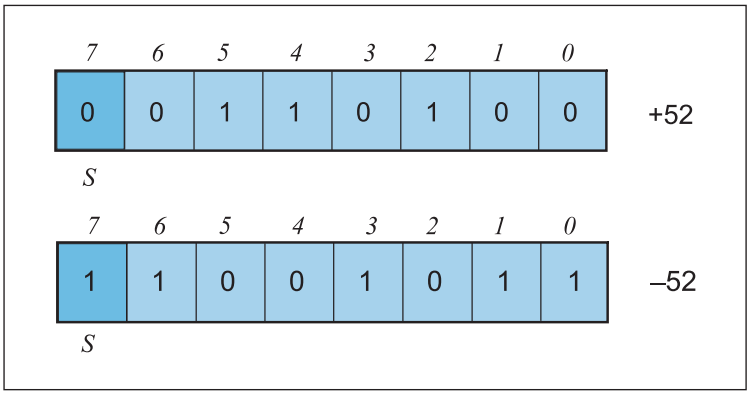


Fig. 3.6. Reprezentarea numerelor +52 şi –52 în cod invers

Codul complementar. În acest cod numerele pozitive au aceeaşi reprezentare ca şi în codul direct şi codul invers. Dacă numărul este negativ, el se scrie mai întîi în codul invers, apoi se adună 1 la cifra cea mai puţin semnificativă (poziţia binară 0).

Drept exemplu, în *figura 3.7* sînt redate reprezentările numerelor +52 şi –52 în cod complementar.

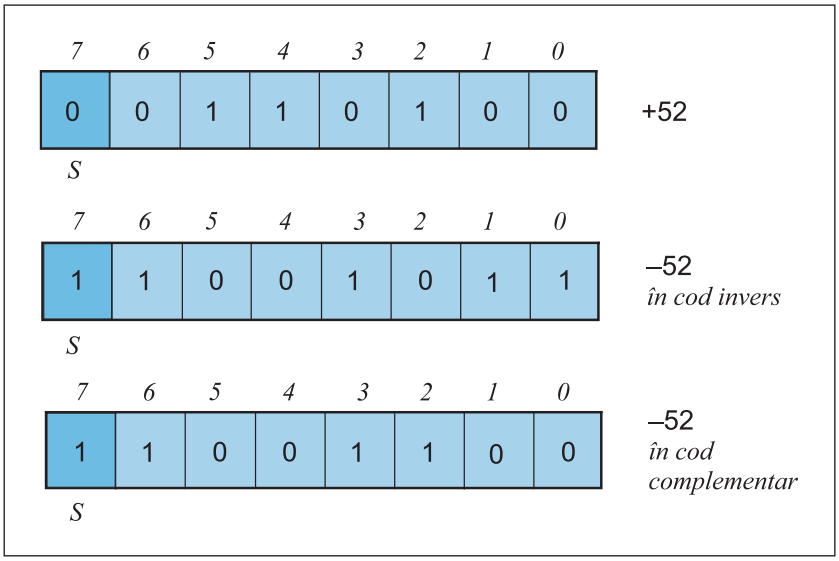


Fig. 3.7. Reprezentarea numerelor +52 şi –52 în cod complementar

Codul complementar este utilizat în marea majoritate a calculatoarelor datorită facilităţilor pe care le oferă la efectuarea operaţiilor aritmetice şi uşurinţei cu care se face verificarea rezultatelor. În acest cod pe n poziţii binare pot fi reprezentate numere întregi din intervalul $[-2^{n-1}, 2^{n-1}-1]$.

Întrebări şi exerciţii

❶ Cum pot fi reprezentate în calculator numerele întregi? Explicaţi cum se scriu numerele negative în cod direct, cod invers şi cod complementar.

❷ Reprezentaţi în cod direct pe 8 poziţii binare:

a)	+12;	*d)*	–12;	*g)*	+21;
b)	–21;	*e)*	–64;	*h)*	–68;
c)	+68;	*f)*	+105;	*i)*	–112.

❸ Reprezentaţi în cod invers pe 8 poziţii 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;	*e)*	–40;	*i)*	+16;
b)	+27;	*f)*	–27;	*j)*	+101;
c)	+109;	*g)*	–109;	*k)*	+111;
d)	–16;	*h)*	–101;	*l)*	–111.

❺ Elaborați 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.

❻ Cum se reprezintă numărul 0 în cod direct, cod invers și cod complementar? Cîte reprezentări are numărul 0 în codurile examinate?

❼ Determinați numărul maxim ce poate fi reprezentat pe n poziții binare în cod direct, cod invers și cod complementar.

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 translatare 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 sînt fracții pure (*fig. 3.8*).

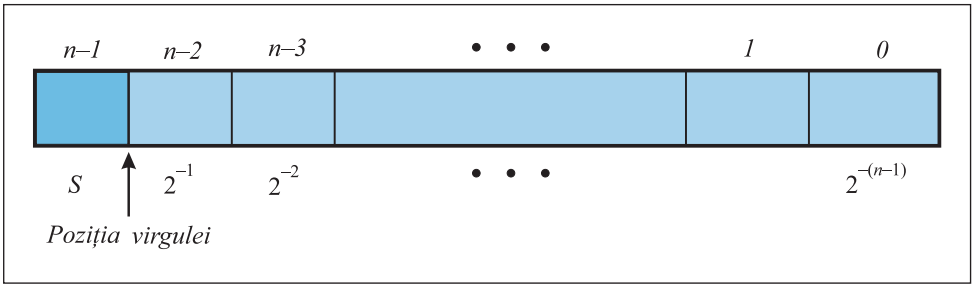


Fig. 3.8. Reprezentarea numerelor reale în virgulă fixă

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

$$0{,}00\ldots0 \le |x| \le 0{,}11\ldots1$$

sau, în sistemul zecimal,

$$0 \le |x| \le 1 - 2^{-(n-1)}.$$

Ca şi 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 poziţii binare în cod direct.

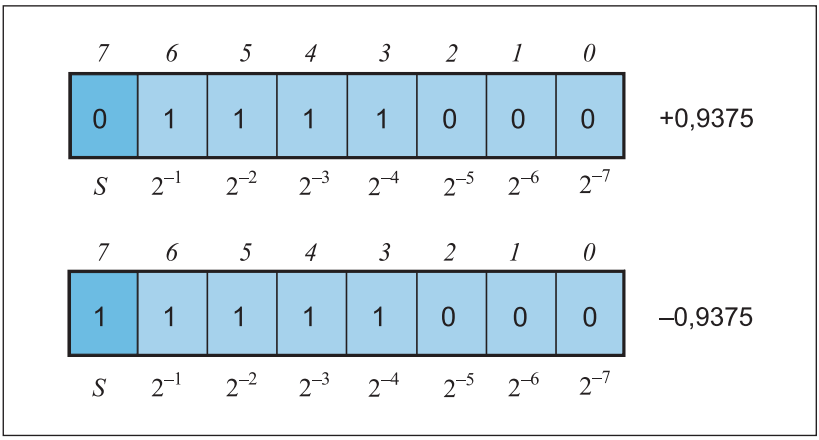


Fig. 3.9. Reprezentarea numerelor +0,9375 şi –0,9375 în virgulă fixă în cod direct

Avantajul principal al reprezentării în virgulă fixă constă în faptul că operaţiile aritmetice cu numere reale pot fi efectuate de dispozitivul aritmetic destinat operării cu numere întregi.

Reprezentarea în virgulă mobilă. Operaţiile în virgulă fixă sînt comode pentru circuitele omogene de date, cînd toate numerele reale sînt subunitare. Însă această reprezentare este ineficientă în calculele ştiinţifice, unde se lucrează simultan cu numere foarte mari şi 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 fracţionare a căror valoare este dată de relaţia:

$$\boldsymbol{x = M \times b^E},$$

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 şi a exponentului pe n poziții binare. În *figura 3.10* este redată reprezentarea în virgulă mobilă în **formatul exponent-mantisă**.

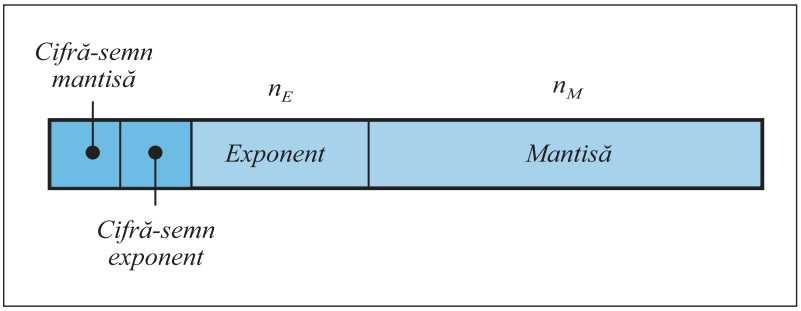


Fig. 3.10. Reprezentarea în virgulă mobilă, formatul exponent-mantisă

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 $n_E = 6...15$ şi $n_M = 24...64$, 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**.

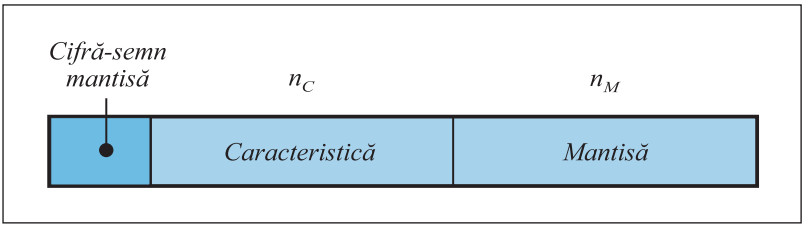


Fig. 3.11. Reprezentarea în virgulă mobilă, formatul caracteristică-mantisă

Caracteristica C constituie o formă de exprimare a exponentului E şi se determină din relația

$$\boldsymbol{C = E + K}.$$

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:

$$C = E + 127,$$

indicînd practic semnul exponentului:

– dacă $C \geq 127$, atunci $E \geq 0$;

– dacă $C < 127$, atunci $E < 0$.

În *tabelul 3.5* sînt 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	$10^{-37}...10^{38}$
Dublă precizie	64	11	52	15 sau 16	$10^{-307}...10^{308}$
Precizie extinsă	80	15	64	19	$10^{-4932}...10^{4932}$

Întrucît 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 numeşte **bit ascuns**. Accentuăm că tehnica bitului ascuns se referă doar la reprezentarea numerelor în memoria calculatorului, nu şi la operațiile efectuate de dispozitivul aritmetic.

Întrebări şi exerciții

❶ Care este deosebirea dintre reprezentarea în virgulă fixă şi reprezentarea în virgulă mobilă?

❷ Reprezentați în virgula 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 sînt avantajele şi dezavantajele reprezentării în virgulă fixă?

❹ Cum se reprezintă numerele reale în virgula mobilă? Care numere reprezentate în virgulă mobilă respectă condiția de normalizare?

❺ Cum se efectuează normalizarea numerelor reprezentate în virgulă mobilă?

❻ 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.

❼ De ce depinde precizia şi domeniul de mărime ale numerelor reprezentate în virgulă mobilă?

❽ Care este deosebirea dintre formatele exponent-mantisă şi caracteristică-mantisă? Cum se calculează caracteristicile numerelor reprezentate în virgulă mobilă?

❾ Calculați caracteristicile numerelor din exercițiul 6. Se consideră că $n_C = 8$.

❿ Explicați termenul „bit ascuns”. Care sînt avantajele acestei reprezentări?

⓫ Cum se reprezintă numerele naturale, întregi şi reale în calculatorul la care lucrați dvs.? Determinați numărul de poziții binare alocate fiecărei reprezentări.

Test de autoevaluare nr. 3

1. Transformați în sistemul zecimal numerele ce urmează:

a) $(821)_9$; *b)* $(1011,121)_3$; *c)* $(341,52)_7$.

2. În care dintre sistemele de numerație *a)*, *b)*, *c)*, *d)* scrierea numărului 284,6 este incorectă? Argumentați răspunsul.

a)	zecimal;	*c)*	binar;
b)	octal;	*d)*	hexazecimal.

3. Elaborați un program PASCAL care transformă în sistemul zecimal numerele scrise în baza *b*, $b < 10$. Baza *b* se citeşte de la tastatură.

4. Transformați în sistemul binar numerele ce urmează:

a) $(13,889)_{10}$; *b)* $(73,542)_8$; *c)* $(A8,74F)_{16}$.

5. Transformați în sistemul octal numerele ce urmează:

a) $(9758,93)_{10}$; *b)* $(1011011011,01011101)_2$; *c)* $(DA86,B1)_{16}$.

6. Transformați în sistemul hexazecimal numerele ce urmează:

a) $(9471,8362)_{10}$; *b)* $(10110111011011,0101001101)_2$; *c)* $(425,376)_8$.

7. Scrieți numerele $(1000001111)_2$, $(132)_8$, $(BB)_{16}$, $(222221)_4$ în ordine descrescătoare.

8. Printre numerele de mai jos există şi numere egale. Scrieți egalitățile respective, de exemplu, $(1010)_2 = (12)_8$.

a)	$(1010)_8$;	*c)*	$(723)_8$;	*e)*	$(21)_{10}$;
b)	$(1D3)_{16}$;	*d)*	$(25)_8$;	*f)*	$(A)_{16}$.

9. Elaborați un program PASCAL pentru conversiunea octal-binară.

10. Calculați în sistemul binar:

a) 110001101 + 10111010; *c)* 111011 × 101;

b) 1101001010 – 101101001; *d)* 101011100 : 110.

11. Reprezentați pe 8 poziții binare numărul natural 125.

12. Reprezentați pe 8 poziții binare în cod complementar numărul întreg –91.

13. Cuvintele binare de mai jos reprezintă numere întregi scrise în cod invers pe 8 poziții binare. Scrieți aceste numere în sistemul zecimal.

a) 11111110; *b)* 00111111; *c)* 11011000.

14. Reprezentați în virgula fixă pe 8 poziții binare în cod direct numărul –0,75.

15. Exprimați în virgulă mobilă următoarele numere:

a) +21,125; *b)* –73,25; *c)* –0,09375.

16. „Cutia neagră” transformă numerele zecimale în numere scrise în baza b. Determinați această bază şi numărul x de la ieşirea din „cutia neagră” în cazul al doilea.

17. Cuvintele binare de mai jos reprezintă numere întregi scrise în cod complementar pe 8 poziții binare. Scrieți aceste numere în sistemul zecimal.

a) 11100110; *b)* 00101101; *c)* 11111000.

18. Precizați intervalul numerelor întregi *N* care se pot reprezenta în cod complementar pe *n* poziții binare:

a) $n = 4$; *b)* $n = 8$; *c)* $n = 16$.

19. Cuvintele binare de mai jos reprezintă numere subunitare în virgulă fixă scrise în cod direct pe 4 poziții binare. Scrieți aceste numere în sistemul zecimal.

a) 1011; *b)* 0001; *c)* 1100.

20. Scrieți în sistemul zecimal cel mai mic (*A*) şi cel mai mare (*B*) numere ce pot fi reprezentate în virgulă fixă în cod direct pe 8 poziții binare.

21. Cuvintele binare de mai jos reprezintă numere reale în virgulă mobilă, baza $b = 2$, formatul exponent-mantisă, scrise în cod direct. Exponentul ocupă $n_E = 2$, iar mantisa $n_M = 4$ poziții binare. Scrieți aceste numere în sistemul zecimal.

a) 11111110; *b)* 00111111; *c)* 11011000.

22. Reprezentați în virgilă mobilă, baza $b = 2$, formatul exponent-mantisă, în cod direct numărul –2,75. Exponentul ocupă $n_E = 2$, iar mantisa $n_M = 4$ poziții binare.

23. Scrieți în sistemul zecimal cel mai mic (*A*) şi cel mai mare (*B*) numere ce pot fi reprezentate în virgulă mobilă, baza $b = 2$, formatul exponent-mantisă, cod direct. Exponentul ocupă $n_E = 2$, iar mantisa $n_M = 4$ poziții binare.

Capitolul 4

ALGEBRA BOOLEANĂ

4.1. Variabile şi expresii logice

Algebra booleană sau **algebra logică** este un compartiment al matematicii în care legile gîndirii – obiectul de studiu al logicii clasice – sînt studiate cu ajutorul metodelor simbolice. Denumirea aceasta a fost dată în onoarea matematicianului englez George Boole, care în lucrarea *The Laws of Thought* („Legile gîndirii"), publicată în 1853, a pus bazele acestei algebre. Mulţi ani algebra booleană a fost considerată drept o simplă curiozitate matematică, fără a i se găsi o utilizare într-un anumit domeniu al ştiinţelor aplicate. Ea a revenit în actualitate odată cu apariţia centralelor telefonice automate şi a calculatoarelor numerice.

Din punct de vedere formal, algebra booleană poate fi definită printr-o mulţime a elementelor {0, 1}, o mulţime a operatorilor elementari {¯, &, ∨} şi printr-un set de postulate. Prin urmare, în algebra booleană orice variabilă nu poate avea decît una din cele două valori posibile, notate simbolic prin 0 şi 1, alte valori neavînd nicio semnificaţie**.**

Variabilele algebrei booleene se notează prin x, y, z, x_1, x_2, ... cu sau fără indicii, iar elementele 0 şi 1 se numesc **constante logice**.

Operatorii elementari ai algebrei booleene au următoarele denumiri:

¯ – negaţia (inversia, operaţia logică *NU*);

& – conjuncţia (produsul logic, operaţia logică *ŞI*);

∨ – disjuncţia (suma logică, operaţia logică *SAU*).

Operatorii elementari se definesc cu ajutorul unor tabele speciale, numite **tabele de adevăr**.

> ***Tabelul de adevăr este un tabel care include toate combinaţiile posibile ale valorilor variabilelor faţă de care este definit operatorul şi rezultatul operaţiei respective.***

În *figura 4.1* sînt prezentate tabelele de adevăr ale negaţiei, conjuncţiei şi disjuncţiei.

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.

Întrucît variabila x poate avea numai valoarea 0 sau 1, tabelul de adevăr al negației conține două rînduri. În cazurile conjuncției și disjuncției tabelele de adevăr conțin patru rînduri, cîte un rînd pentru fiecare din combinațiile posibile 00, 01, 10 și 11 ale variabilelor x și y.

Variabilele și constantele logice, reunite cu ajutorul operatorilor $\bar{}$, & și $\vee$ formează **expresii logice**, de exemplu:

1) $x \,\&\, y \vee z$;

2) $x \vee y \,\&\, z$;

3) $\bar{x} \,\&\, y \vee z$;

4) $1 \,\&\, x \vee y$;

5) $0 \vee x \vee y$.

6) $1 \vee 0$.

Valorile expresiilor logice pot fi calculate utilizînd 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 întîi se execută operația &, iar după aceea operația $\vee$. În expresia

$$x \,\&\, y \vee \bar{z}$$

se execută negația, în continuare conjuncția și, în sfîrșit, disjuncția.

Ordinea executării operațiilor logice poate fi schimbată cu ajutorul parantezelor „(” și „)”. Evident, în primul rînd, 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 şi în forma:

$$\bar{x}y \vee z,$$
$$(x \vee y)(\bar{x}\ \vee z).$$

Întrebări şi exerciții

❶ Ce valori poate avea o variabilă logică?

❷ Care sînt operatorii elementari ai algebrei booleene şi cum se definesc ei?

❸ Memorizați tabelele de adevăr ale negației, conjuncției şi disjuncției.

❹ Cum se formează expresiile logice? Care este prioritatea operațiilor logice?

❺ Explicați rolul parantezelor din componența expresiilor logice.

❻ Alcătuiți tabelele de adevăr ale următoarelor expresii logice:

a) $\bar{x}y$; *c)* $\bar{x} \vee y$;

b) $\bar{x}\bar{y}$; *d)* $\overline{x \vee y}$;

e) $\bar{\bar{x}}$;
f) $\bar{x}x$;
g) $\overline{xy}$;
h) $\bar{x} \vee x$;
i) xyz;
j) $x \vee y \vee z$;
k) $xy \vee z$;
l) $x\,(y \vee z)$;
m) $\bar{x} \vee y \vee z$;
n) $\bar{x}\,(y \vee z)$.

❼ Elaborați un program PASCAL care introduce valorile logice ale variabilelor x, y, z și afișează valorile uneia dintre expresiile ce urmează:

a) $\bar{x}$;
b) xy
c) $x \vee y$;
d) $\overline{(x \vee y)}\,z$;
e) $\overline{xy} \vee z$;
f) $x \vee y \vee z$;
g) $xy \vee z$;
h) $\overline{x \vee y \vee z}$;
i) $x \vee xy$;
j) $\bar{x} \vee y \vee z$.

❽ Elaborați un program PASCAL care afișează tabelele de adevăr ale conjuncției și disjuncției.

❾ Elaborați un algoritm repetitiv care formează toate combinațiile posibile ale valorilor variabilelor x, y și z. Afișați combinațiile respective pe ecran.

❿ Pentru fiecare dintre expresiile logice ce urmează elaborați cîte un program PASCAL care alcătuiește tabelul de adevăr respectiv:

a) $x \vee \bar{x}$;
b) $x\bar{x}$;
c) $\overline{x \vee y}$;
d) $\overline{xy}$;
e) $\bar{x} \vee y$;
f) $\bar{x}y$;
g) $x \vee y \vee z$;
h) xyz;
i) $(x \vee y)(\bar{x} \vee y)$;
j) $(x \vee y)(x \vee \bar{y})$.

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 $x_1, x_2, ..., x_n$ un grup arbitrar de variabile booleene, unde $n = 1, 2, 3, ...$. Întrucît fiecare variabilă booleană poate avea numai valorile 0 sau 1, numărul tuturor combinațiilor posibile ale valorilor variabilelor $x_1, x_2, ..., x_n$ este de 2^n.

Firește, pentru $n = 1$ avem 2 combinații (0 și 1); pentru $n = 2$ sînt $2^2 = 4$ 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 $y = f(x_1, x_2, ..., x_n)$ ***este o aplicație care pune în corespondență fiecărei combinații de valori ale variabilelor*** $x_1, x_2, ..., x_n$ ***valoarea 0 sau 1 a variabilei y.***

Variabilele $x_1, x_2, ..., x_n$ se numesc **variabile independente** sau **argumente**, iar variabila y – **variabilă dependentă** sau **funcție** de argumente $x_1, x_2, ..., x_n$.

Prin urmare, **domeniul de definiție** al funcției $y = f(x_1, x_2, ..., x_n)$ este mulțimea tuturor combinațiilor posibile

0 0 ... 0
0 0 ... 1
...
1 1 ... 1

ale valorilor argumentelor $x_1, x_2, ..., x_n$, 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 $y = f(x_1, x_2, ..., x_n)$ ***este un tabel care include toate combinațiile posibile ale valorilor argumentelor*** $x_1, x_2, ..., x_n$ ***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 trei variabile

Tabelul include $2^3 = 8$ rînduri 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 $x_1, x_2, ..., x_n$.

De exemplu,

1) $y = x$;
2) $y = x_1 x_2$;
3) $y = \bar{x}_1 x_2 \vee x_3$;
4) $y = \overline{x_1 x_2 \vee x_3}$.

Natural, cunoscînd formula unei funcţii logice, poate fi calculat tabelul ei de adevăr. De exemplu, tabelul de adevăr al funcţiei

$$y = \overline{x_1 x_2 \vee x_3}$$

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 notaţii, în *figura 4.3* din paragraful 4.1.

Există mai multe **metode grafice de definire a funcţiilor logice**. Aceste metode se bazează pe diagramele utilizate în teoria mulţimilor şi se studiază în cursurile avansate de informatică.

Întrebări şi exerciţii

❶ Formulaţi definiţia noţiunii „funcţie logică”.

❷ Care sînt domeniul de definiţie şi domeniul de valori ale unei funcţii logice?

❸ Prin ce metode poate fi definită o funcţie logică de n variabile?

❹ Cum se alcătuieşte tabelul de adevăr al unei funcţii de n variabile? Cîte rînduri conţine acest tabel?

❺ Cum poate fi alcătuit tabelul de adevăr al unei funcţii logice atunci cînd se cunoaşte formula ei?

❻ Funcţia logică de 3 variabile $y = f(x_1, x_2, x_3)$ este definită prin tabelul de adevăr din *figura 4.4.* Numiţi combinaţiile valorilor argumentelor x_1, x_2, x_3 pentru care funcţia dată are valoarea $y = 1$. Numiţi combinaţiile respective pentru valoarea funcţiei $y = 0$.

❼ Alcătuiţi tabelele de adevăr ale următoarelor funcţii logice:

a) $y = x$;

b) $y = \bar{x}$;

c) $y = x_1 x_2$;

d) $y = x_1 \vee x_2$;

e) $y = \overline{x_1 x_2}$;

f) $y = \overline{x_1 \vee x_2}$;

g) $y = \overline{x_1 x_2 x_3}$;

h) $y = x_1(x_2 \vee \bar{x}_3)$;

i) $y = \bar{x}_1 \vee x_2 x_3$;

j) $y = x_1 \vee \overline{x_2 x_3}$.

❽ Elaboraţi un program PASCAL care introduce valorile logice ale variabilelor x_1, x_2, x_3, x_4 şi afişează valorile uneia dintre funcţiile ce urmează:

a) $y = x_1 x_2 \vee x_3 x_4$;

b) $y = (x_1 \vee x_2)(x_3 \vee x_4)$;

c) $y = \overline{x_1 x_2 x_3 x_4}$;

d) $y = \overline{x_1 \vee x_2 \vee x_3 \vee x_4}$;

e) $y = \overline{x_1 x_2} \vee \overline{x_3 x_4}$;

f) $y = \overline{(x_1 \vee x_2)(x_3 \vee x_4)}$;

g) $y = x_1 \bar{x}_2 \vee x_3 \bar{x}_4$;

h) $y = \bar{x}_1 x_2 \vee \bar{x}_3 x_4$;

i) $y = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4$;

j) $y = x_1 x_2 x_3 x_4 \vee \bar{x}_2 x_3 x_4$;

k) $y = x_1 \vee x_2 x_3 \vee x_4$;

l) $y = x_1 \vee x_2 \vee x_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$.

❾ Elaborați un program PASCAL care alcătuiește tabelul de adevăr al uneia dintre funcțiile ce urmează:

a) $y = x$;
b) $y = \bar{x}$;
c) $y = x_1 x_2$;
d) $y = x_1 \vee x_2$;
e) $y = \overline{x_1 x_2}$;
f) $y = \bar{x}_1 \vee \bar{x}_2$;
g) $y = \overline{x_1 \vee x_2}$;
h) $y = \bar{x}_1 \bar{x}_2$;
i) $y = \overline{x_1 x_2 x_3}$;
j) $y = \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$;
k) $y = \overline{x_1 \vee x_2 \vee x_3}$;
l) $y = \bar{x}_1 \bar{x}_2 \bar{x}_3$;
m) $y = x_1 \vee x_2 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$;
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 $x_1, x_2, ..., x_n$. Apare întrebarea, cîte funcții logice de n variabile există în algebra booleană? **Numărul funcțiilor logice posibile** poate fi determinat prin următoarele raționamente.

Întrucît 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. Întrucît tabelul de adevăr are 2^n rînduri, există

$$m = 2^{2^n}$$

funcții logice de n variabile. Funcțiile respective se notează prin y_j, $j = 0, 1, ..., m-1$.

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, **constanta 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
1	0	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} sînt 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 *ŞI*** sau **conjuncția**.

Funcția y_{14} poate fi scrisă în forma:

$$y_{14} = f(x_1, x_2) = x_1 \vee x_2.$$

Prin urmare, funcția y_{14} va avea denumirea **funcția logică *SAU*** sau **disjuncția**.

Funcțiile logice NU, ŞI, SAU, induse de operatorii elementari, respectiv, $\bar{}$, &, $\vee$ se numesc funcții logice elementare.

Din *figura 4.6* se observă că

$$y_1 = \overline{x_1 \vee x_2}.$$

Funcția dată se numeşte **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 numeşte **funcția logică *ŞI-NU***.

Funcția

$$y_9 = f(x_1, x_2) = \bar{x}_1 \bar{x}_2 \vee x_1 x_2$$

are valoarea 1 numai cînd $x_1 = x_2 = 0$ sau $x_1 = x_2 = 1$. Această funcție se numeşte **funcția logică *COINCIDENȚĂ*** sau **echivalență** şi se notează prin simbolul „≡”.

Analizînd tabelul din *figura 4.6*, mai observăm că

$y_3 = f(x_1, x_2) = \bar{x}_1$ (negația variabilei x_1);

$y_{12} = f(x_1, x_2) = x_1$ (repetarea variabilei x_1);

$y_5 = f(x_1, x_2) = \bar{x}_2$ (negația variabilei x_2);

$y_{10} = f(x_1, x_2) = x_2$ (repetarea variabilei x_2).

Pentru a fi mai uşor memorizate, în *figura 4.7* sînt prezentate tabelele de adevăr ale funcțiilor logice *NU, ŞI, SAU, ŞI-NU, SAU-NU* şi *COINCIDENȚĂ*.

NU

x	$\bar{x}$
0	1
1	0

ŞI

x_1	x_2	x_1x_2
0	0	0
0	1	0
1	0	0
1	1	1

SAU

x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

ŞI-NU

x_1	x_2	$\overline{x_1x_2}$
0	0	1
0	1	1
1	0	1
1	1	0

SAU-NU

x_1	x_2	$\overline{x_1 \vee x_2}$
0	0	1
0	1	0
1	0	0
1	1	0

COINCIDENŢĂ

x_1	x_2	$x_1 \equiv x_2$
0	0	1
0	1	0
1	0	0
1	1	1

Fig. 4.7. Funcţiile logice frecvent utilizate

În mod similar pot fi studiate funcţiile logice de 3 variabile, numărul cărora este de $m = 2^{2^3} = 2^8 = 256$; funcţiile de 4 variabile, numărul cărora este de $m = 2^{2^4} = 2^{16} = 65\,536$ etc. Se observă că, deşi este finit, numărul funcţiilor booleene posibile este enorm. S-a demonstrat însă că **orice funcţie logică de *n* variabile, $n \geq 2$, poate fi exprimată printr-o formulă care include numai operatorii elementari $\bar{}$, &, $\vee$.** Această proprietate facilitează realizarea tehnică a dispozitivelor destinate calculării funcţiilor logice cu un număr arbitrar de argumente.

Întrebări şi exerciţii

❶ Determinaţi numărul funcţiilor logice de 5 şi de 6 variabile.
❷ Numiţi funcţiile logice elementare şi alcătuiţi tabelele respective de adevăr.
❸ Memorizaţi tabelele de adevăr ale funcţiilor logice frecvent utilizate *NU*, *ŞI*, *SAU*, *ŞI-NU*, *SAU-NU* şi *COINCIDENŢĂ*.
❹ Elaboraţi un program care va afişa pe ecran tabelul de adevăr al uneia dintre funcţiile logice y_0, y_1, y_2, y_3 de 2 variabile.
❺ Elaboraţi un program care va afişa pe ecran tabelul de adevăr al uneia dintre funcţiile logice y_j de n variabile.

Test de autoevaluare nr. 4

1. Alcătuiţi tabelul de adevăr al expresiei logice $x \vee y\bar{z}$.

2. Care dintre expresiile logice ce urmează sînt egale? Amintim că două expresii logice sînt egale dacă valorile lor coincid pentru toate combinaţiile posibile ale valorilor variabilelor respective.

a) $(\bar{x} \vee \bar{y})(x \vee y)$;
b) $xy \vee \overline{x\bar{y}}$;
c) $(x \vee y) \vee \overline{x\bar{y}}$;
d) $x\bar{y} \vee \bar{x}y$.

3. Scrieți toate combinațiile posibile de valori ale variabilelor x, y, z pentru care valoarea expresiei $xy \vee \bar{z}$ este egală cu 1.

4. Elaborați un program PASCAL care citeşte de la tastatură valorile variabilelor logice x, y, z şi afişează pe ecran valoarea expresiei $x\bar{y} \vee z$.

5. Elaborați un program PASCAL care afişează pe ecran tabelul de adevăr al expresiei $\bar{x}\bar{y} \vee z$.

6. Indicați domeniul de definiție şi domeniul de valori ale funcției logice $y = x_1(x_2 \vee \bar{x}_3)$.

7. Alcătuiți tabelul de adevăr al funcției logice $y = x_1(\bar{x}_2 \vee x_3)$.

8. Elaborați un program PASCAL care citeşte de la tastatură valorile variabilelor logice x_1, x_2, x_3 şi afişează pe ecran valoarea funcției $y = x_1(\bar{x}_2 \vee \bar{x}_3)$.

9. Elaborați un program PASCAL care afişează pe ecran tabelul de adevăr al funcției logice $y = \bar{x}_1(x_2 \vee x_3)$.

10. Determinați numărul funcțiilor logice de 5 variabile.

11. Alcătuiți tabelele de adevăr ale funcțiilor logice *ŞI-NU* şi *SAU-NU*.

12. Care dintre funcțiile logice ce urmează sînt egale? Amintim că două funcții logice sînt egale dacă valorile lor coincid pentru toate combinațiile posibile ale valorilor variabilelor independente.

a) $y = \overline{x_1 \vee x_2}$; *c)* $y = \bar{x}_1\bar{x}_2$;

b) $y = \overline{x_1 x_2}$; *d)* $y = \bar{x}_1 \vee \bar{x}_2$.

13. Funcția logică de trei variabile $y = f(x_1, x_2, x_3)$ este definită cu ajutorul următorului tabel de adevăr:

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Selectați din expresiile ce urmează formula care, de asemenea, defineşte această funcție:

a) $y = x_1 \vee x_2 \vee x_3$; *c)* $y = \overline{x_1 x_2 x_3}$;

b) $y = \overline{x_1 \vee x_2 \vee x_3}$; *d)* $y = \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Capitolul 5

CIRCUITE LOGICE

5.1. Circuite logice elementare

Circuitul logic este un dispozitiv destinat calculării funcţiilor logice. Pentru a realiza circuitele logice, e necesar ca valorile binare **0** şi **1** ale argumentelor şi funcţiilor respective să fie reprezentate prin valorile unor mărimi fizice, de exemplu: presiune, temperatură, tensiune sau curent electric, flux luminos etc. În funcţie de mărimile fizice utilizate, deosebim dispozitive logice mecanice, pneumatice, hidraulice, electromecanice, electronice, optice etc. În dispozitivele hidraulice şi pneumatice valorile logice **0** sau **1** pot fi reprezentate prin valorile mici şi, respectiv, mari ale presiunii fluidului, în dispozitivele electromecanice şi electronice – prin prezenţa sau absenţa curentului electric, prin niveluri de tensiune etc.

Pentru o înţelegere clară a principiilor de funcţionare a dispozitivelor logice, vom studia mai întîi circuitele cu contacte. Componentele de bază ale acestor circuite sînt **elementele de comutare** – contactele electrice normal deschise şi contactele electrice normal închise.

În cazul **contactelor normal deschise**, circuitul electric este deschis dacă contactele nu sînt acţionate şi – închis la acţionarea lor. În cazul **contactelor normal închise**, circuitul electric este închis dacă contactele nu sînt acţionate şi – deschis la acţionarea lor (*fig. 5.1*).

	Neacţionate	*Acţionate*
Contacte normal deschise		
Contacte normal închise		

Fig. 5.1. Contacte normal deschise şi normal închise

De exemplu, contactele unui întrerupător electric uzual sînt contacte normal deschise, iar contactele butonului de pauză al magnetofonului sînt contacte normal închise.

În circuitele cu contacte, valorile logice ale argumentelor sînt reprezentate prin stările contactelor electrice respective. Valorii logice **1** îi corespunde starea „*contactul este acţionat*”, iar valorii logice **0** îi corespunde starea „*contactul este neacţionat*”.

Circuitul electric care realizează **funcția logică *NU*** şi simbolul utilizat sînt prezentate în *figura 5.2*.

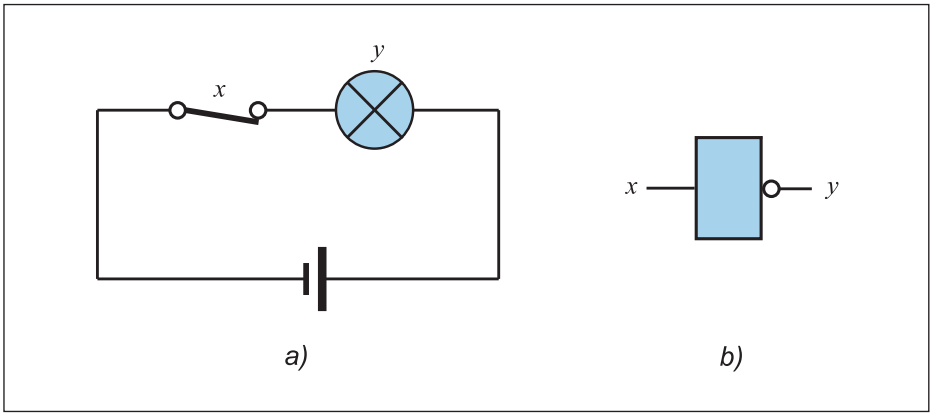


Fig. 5.2. Circuit cu contacte pentru realizarea funcției logice *NU* (*a*) şi simbolul utilizat (*b*)

Argumentul x este materializat prin contactul normal închis, iar valorile variabilei dependente y sînt 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 *ŞI* se realizează prin conectarea în serie a contactelor electrice. Circuitul electric care realizează funcția *ŞI* de două variabile şi simbolul utilizat sînt prezentate în *figura 5.3*.

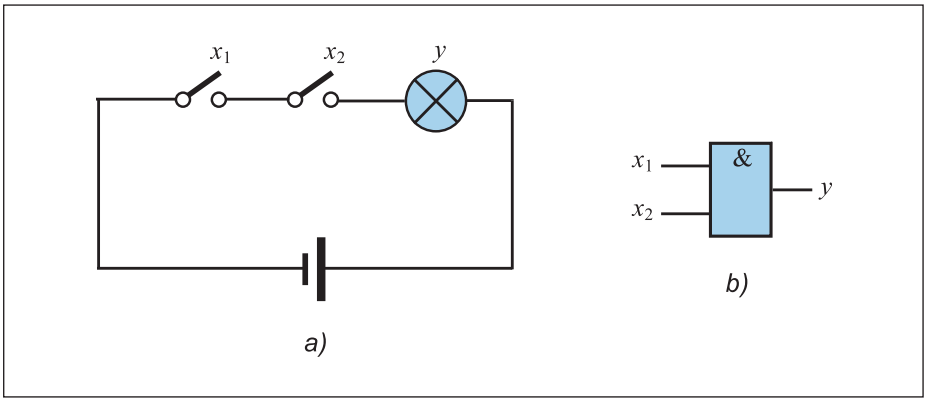


Fig. 5.3. Circuit cu contacte pentru realizarea funcției logice *ŞI* (*a*) şi simbolul utilizat (*b*)

Variabilele x_1 şi x_2 sînt materializate prin cele două contacte normal deschise, iar valorile variabilei y prin bec. Se observă că becul va fi aprins (y=1), numai dacă ambele contacte normal deschise sînt acționate (x_1=1 şi x_2=1).

Funcția logică *SAU* se realizează prin conectarea în paralel a contactelor electrice. Circuitul respectiv este prezentat în *figura 5.4*.

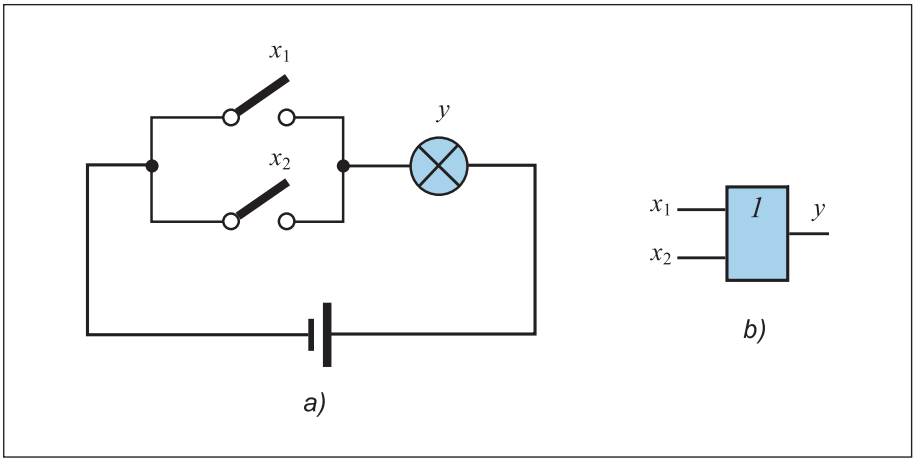


Fig. 5.4. Circuite cu contacte pentru realizarea funcției logice *SAU* (*a*) şi simbolul utilizat (*b*)

Se constată că becul va fi aprins (y=1), dacă cel puțin unul dintre cele două contacte normal deschise este acționat (x_1=1 sau x_2=1).

Întrucît viteza de închidere-deschidere a contactelor electrice este foarte mică, în calculatoarele moderne valorile **0, 1** sînt 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 obişnuit, care într-o stare conduce curentul (este închis), iar în alta – nu (este deschis). Spre deosebire însă de întrerupătorul obişnuit, 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) şi schemele echivalente în regim de comutație.

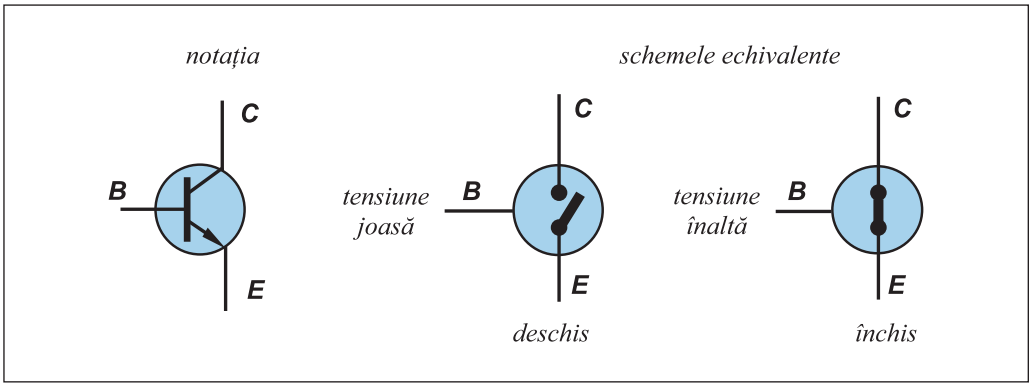


Fig. 5.5. Tranzistorul *n-p-n*

Tranzistorul *n-p-n* are trei terminale: emitorul *E*, baza *B* şi colectorul *C*. În regim de comutaţie, emitorul şi colectorul pot fi consideraţi drept contacte care se închid sau se deschid cu ajutorul unei tensiuni aplicate la bază. Menţionăm că tranzistoarele moderne permit efectuarea a $10^6 – 10^9$ închideri–deschideri pe secundă. Ca şi în cazul contactelor electrice studiate mai sus, utilizarea diferitor tipuri de tranzistoare şi conectarea lor în serie sau în paralel permite realizarea funcţiilor logice *NU*, *ŞI*, *SAU*.

Circuitele destinate calculării funcţiilor logice frecvent utilizate se numesc circuite logice elementare sau porţi logice.

Simbolurile utilizate pentru notarea porţilor logice sînt reprezentate în *figura 5.6*.

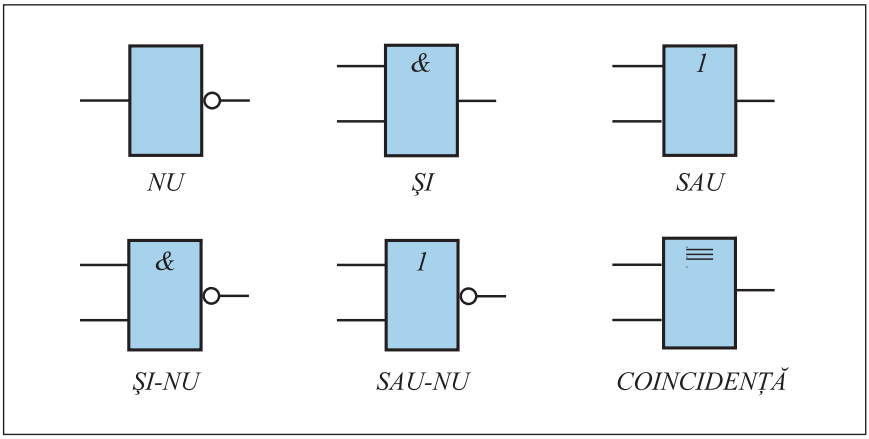


Fig. 5.6. Simbolurile porţilor logice

E cunoscut faptul că orice funcţie logică poate fi exprimată printr-o formulă care include numai simbolurile operatorilor elementari $\bar{}$, &, $\vee$. Prin urmare, **orice funcţie logică cu un număr arbitrar de argumente poate fi materializată prin conectarea porţilor logice *NU*, *ŞI*, *SAU*.** De exemplu, funcţia

$$y = x_1x_2 \vee \bar{x}_2x_3$$

poate fi realizată cu ajutorul următoarelor porţi logice:

– o poartă *NU* pentru calcularea $\bar{x}_2$;

– două porţi logice *ŞI* pentru calcularea conjuncţiilor x_1x_2 şi $\bar{x}_2x_3$;

– o poartă *SAU* pentru calcularea disjuncţiei $x_1x_2 \vee \bar{x}_2x$;

Schema circuitului logic pentru calcularea funcţiei respective este prezentată în *figura 5.7*.

Întrebări şi exerciţii

❶ Cum pot fi reprezentate valorile binare **0** şi **1**?

❷ Cum funcţionează contactele normal deschise şi contactele normal închise?

❸ Care este reprezentarea valorilor binare **0** şi **1** în circuitele cu contacte?

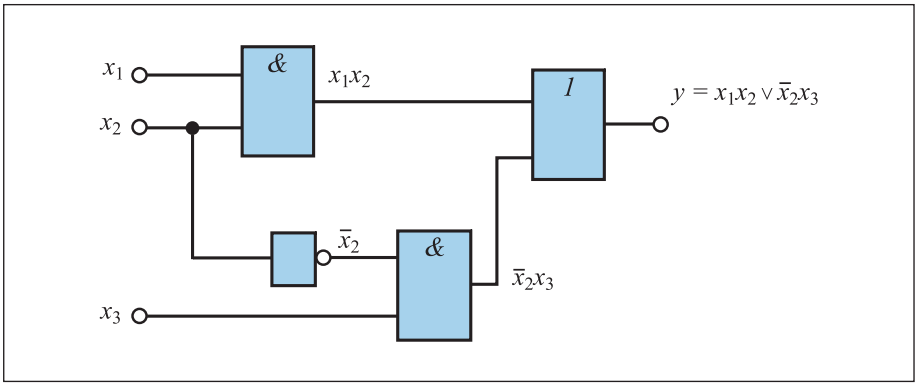


Fig. 5.7. Circuitul logic pentru realizarea funcției $y = x_1x_2 \vee \bar{x}_2x_3$

❹ Utilizînd trusa din laboratorul de fizică, montaţi circuitele din *figurile 5.2*, *5.3* şi *5.4*. Verificaţi tabelele de adevăr ale funcţiilor realizate de circuitele în cauză.

❺ Cum se realizează funcţiile *NU*, *ŞI*, *SAU* în cazul circuitelor cu contacte electrice?

❻ Care este rolul elementelor de comutare la realizarea circuitelor logice?

❼ Care este rolul tranzistorului în calculatoarele moderne?

❽ Memorizaţi simbolurile porţilor logice. Explicaţi cum se utilizează porţile logice la realizarea funcţiilor logice arbitrare.

❾ Utilizînd porţile *NU*, *ŞI*, *SAU,* elaboraţi circuitele logice pentru calcularea următoarelor funcţii:

a) $y = x_1x_2x_3$;
b) $y = x_1 \vee x_2 \vee x_3$;
c) $y = \bar{x}_1x_2x_3$;
d) $y = \bar{x}_1 \vee x_2 \vee x_3$;
e) $y = x_1x_2 \vee x_3x_4$;
f) $y = (x_1 \vee x_2)(x_3 \vee x_4)$;
g) $y = \overline{x_1x_2}$;
h) $y = \overline{x_1 \vee x_2}$;
i) $y = x_1x_2 \vee \overline{x_2x_3}$;
j) $y = (x_1 \vee x_2)(\overline{x_2 \vee x_3})$;
k) $y = x_1x_2 \vee \bar{x}_1x_3 \vee x_3x_4$;
l) $y = \bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3$;
m) $y = \bar{x}_1x_2 \vee x_1\bar{x}_2$;
n) $y = x_1x_2 \vee \bar{x}_1\bar{x}_2$;
o) $y = x_1(x_2 \vee x_3 \vee x_4)$;
p) $y = x_1 \vee \overline{x_2x_3x_4}$.

❿ Releul electromagnetic este un dispozitiv cu care se comandă închiderea sau deschiderea contactelor electrice. Contactele respective sînt acţionate de un electromagnet. Cum poate fi utilizat releul pentru realizarea funcţiilor logice *NU*, *ŞI*, *SAU*?
Cu piesele din trusa de fizică asamblaţi un releu electromagnetic şi verificaţi soluţiile propuse de dvs.

⓫ Reprezentînd valorile binare ale variabilelor de ieşire prin prezenţa (valoarea **1**) sau absenţa (valoarea **0**) a fluidului, elaboraţi circuitele hidraulice pentru realizarea funcţiilor logice *NU*, *ŞI*, *SAU*. Montaţi instalaţiile respective utilizînd robinetele şi furtunurile din trusa de chimie. Verificaţi tabelele de adevăr ale funcţiilor logice realizate.

5.2. Clasificarea circuitelor logice

Circuitele logice se clasifică în două categorii: circuite combinaționale şi circuite secvențiale.

Într-un **circuit combinațional** valorile variabilelor de ieşire sînt determinate în orice moment de combinația valorilor variabilelor de intrare conform funcțiilor logice ale circuitului. Într-un **circuit secvențial** valorile variabilelor de ieşire depind nu numai de combinațiile valorilor variabilelor de intrare, dar şi de consecutivitatea aplicării lor. Altfel spus, circuitele combinaționale reprezintă circuite logice lipsite de elemente de memorie, iar circuitele secvențiale includ şi 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*.

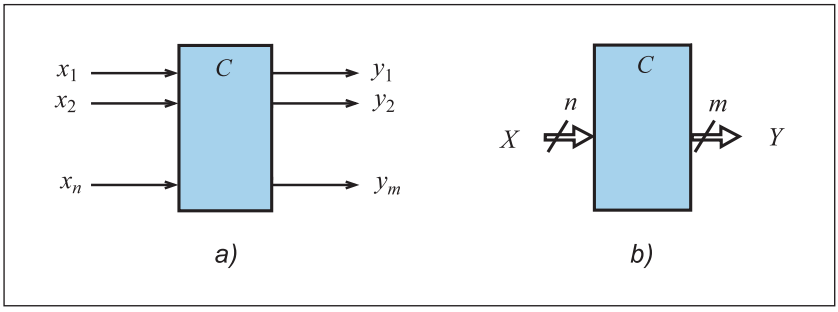


Fig. 5.8. Schema-bloc a unui circuit logic combinațional:
a – detaliat; *b* – generalizat

Circuitul are n variabile de intrare: $X = < x_1, x_2, ..., x_n >$, şi m variabile de ieşire: $Y = < y_1, y_2, ..., y_m >$. Conexiunile destinate transmiterii valorilor unui grup de variabile se reprezintă prin linii duble. Dacă e necesar, numărul de variabile se indică lîngă 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 ieşirile 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 şi, în special, adunarea şi 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 combinaţional destinat adunării a două cifre binare. Tabelul de adevăr care pune în evidenţă funcţionarea unui semisumator rezultă din regula de adunare a două cifre binare şi este prezentat în *figura 5.9*. Aici a şi 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 funcţiile de ieşire s şi t:

$$s = \bar{a}b \vee a\bar{b};$$

$$t = ab.$$

Schema care realizează funcţiile s, t şi simbolul utilizat sînt prezentate în *figura 5.10*.

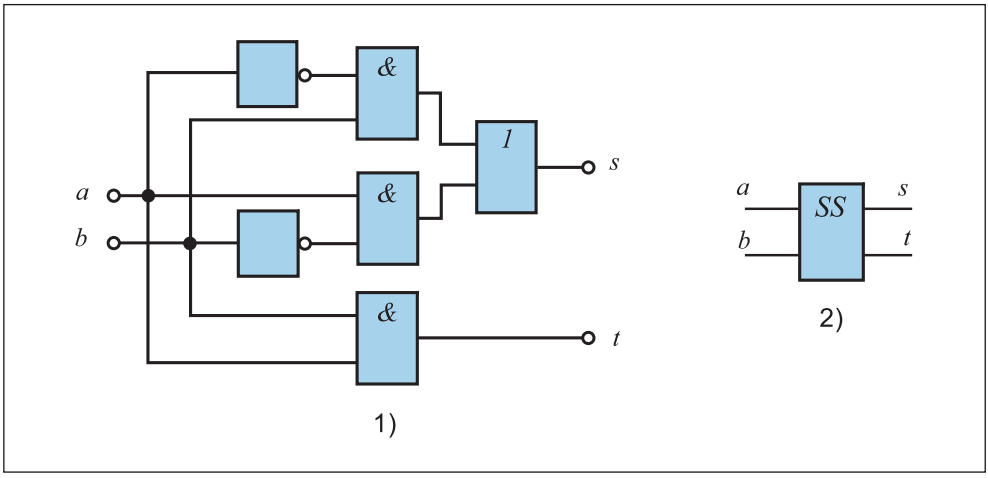


Fig. 5.10. Schema semisumatorului (1) şi simbolul utilizat (2)

Să luăm două numere binare

$$A = a_{n-1}\, a_{n-2} \ldots a_j \ldots a_0$$

şi

$$B = b_{n-1}\, b_{n-2} \ldots b_j \ldots b_0,$$

unde a_j şi b_j reprezintă cifrele binare din rangul (poziţia) j. La însumarea cifrelor a_j şi b_j din rangul j trebuie să se ia în considerare şi cifra de transport t_{j-1} de la rangul inferior j–1:

$$
\begin{array}{r}
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad t_{j-1} \\
a_{n-1}\ a_{n-2}\ \ldots\ a_j\ \ldots\ a_0 \\
+ \\
b_{n-1}\ b_{n-2}\ \ldots\ b_j\ \ldots\ b_0.
\end{array}
$$

Se obţine astfel un circuit combinaţional 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 şi SS_2 (*fig. 5.11*).

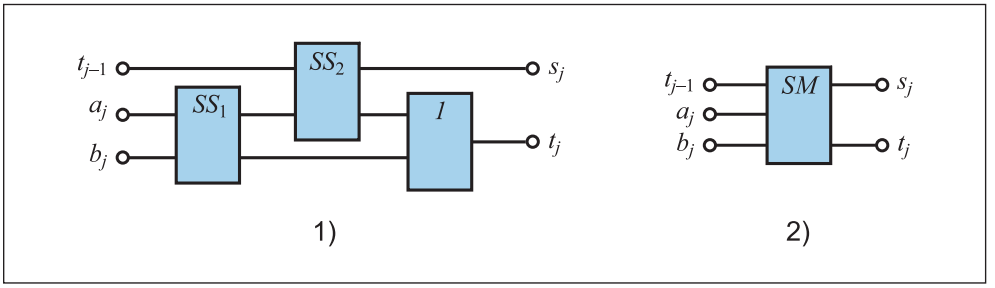


Fig. 5.11. Schema sumatorului elementar (1) şi simbolul utilizat (2)

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 reuneşte transporturile intermediare de la ieşirile respective ale semisumatoarelor SS_1 şi SS_2.

Suma numerelor binare *A* şi *B* se calculează cu ajutorul unui circuit combinaţional denumit **sumator**. Un sumator poate fi realizat prin conectarea a *n* sumatoare elementare (*fig.5.12*).

Sumatorul elementar SM_0 corespunzător cifrei celei mai puţin semnificative poate fi înlocuit cu un semisumator, deoarece pentru această poziţie nu există un transport de la rangul precedent. Transportul de la ieşirea sumatorului SM_{n-1} al rangului cel mai semnificativ este folosit pentru a indica **depăşirea capacităţii** sumatorului de *n* biţi.

Din analiza *figurilor 5.10*, *5.11* şi *5.12* rezultă că un dispozitiv complex – sumatorul de *n* biţi este realizat prin reunirea a unor dispozitive mult mai simple, adică a *n* sumatoare elementare. Fiecare sumator elementar, la rîndul lui, este realizat prin reunirea a cîte două semisumatoare şi 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 – porţi logice;
nivelul 3 – semisumatoare, sumatoare elementare etc.;
nivelul 4 – sumatoare, scăzătoare etc.;
nivelul 5 – unităţi aritmetice, unităţi de comandă etc.

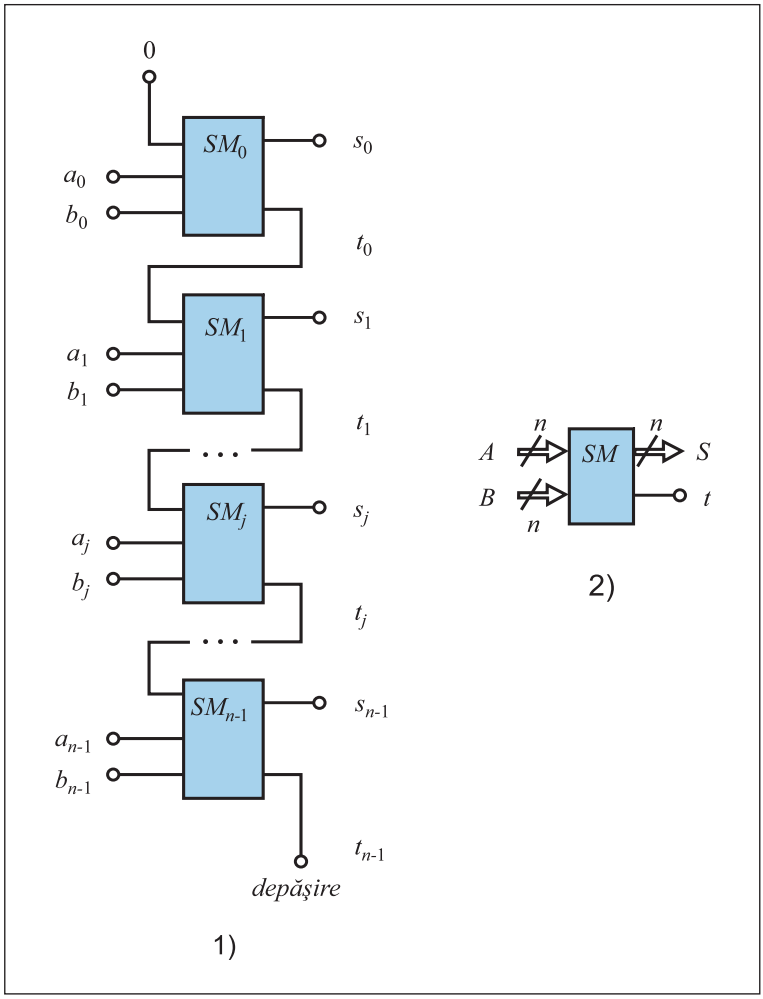


Fig. 5.12. Schema sumatorului (1) şi simbolul utilizat (2)

Componentele de un nivel ierarhic inferior sînt utilizate în calitate de „cubuşoare" elementare pentru realizarea componentelor de un nivel ierarhic superior.

Teoretic, dispozitivele unui calculator numeric pot fi elaborate şi 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 funcţiile de ieşire prin formule şi să reunim porţile 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 biţi, tabelul respectiv de adevăr ar conţine 2^{2n} rînduri. Pentru $n = 16$ obţinem $2^{2 \cdot 16} = 2^{32} \approx 10^9$ rînduri. Evident, formulele funcţiilor de ieşire ale sumatorului de 16 biţi practic nu mai pot fi scrise. Prin urmare, sîntem nevoiţi să aplicăm metoda de proiectare ierarhică şi să realizăm sumatorul printr-o reunire de n sumatoare elementare.

Aplicînd metoda de proiectare ierarhică, într-un mod similar pot fi elaborate circuitele combinaționale destinate scăderii numerelor binare: **semiscăzătorul**, **scăzătorul elementar** şi **scăzătorul**.

Întrebări şi exerciții

❶ Care este destinația semisumatorului? Dar a sumatorului elementar? Şi în fine a sumatorului de n biți?

❷ Alcătuiți tabelul de adevăr al sumatorului elementar. Tabelul va conține cinci coloniță: trei pentru intrările a_j, b_j, t_{j-1} şi două pentru ieşirile s_j, t_j.

❸ Elaborați un program PASCAL care simulează funcționarea sumatorului elementar. Cifrele binare a_j, b_j şi cifra de transport t_{j-1} de la rangul inferior se citesc de la tastatură, iar cifra-sumă s_j şi cifra de transport t_j către rangul superior se afişează pe ecran.

❹ Explicați esența metodei de proiectare ierarhică a dispozitivelor unui calculator numeric. Este oare obligatorie aplicarea acestei metode? Argumentați răspunsul dvs.

❺ Cîte porți logice *NU*, *ŞI*, *SAU* va conține un sumator de 16 biți? Dar de 32 de biți?

❻ **Semiscăzătorul** este un circuit combinațional destinat scăderii a două cifre binare. Circuitul respectiv are intrările a, b şi ieşirile 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 şi elaborați schema semiscăzătorului.

❼ **Scăzătorul elementar** este un circuit combinațional capabil de a calcula diferența d_j şi împrumutul i_j către rangul imediat superior, dacă se introduc la intrare descăzutul a_j, scăzătorul b_j şi împrumutul i_{j-1} de la rangul anterior. Aplicînd metoda de proiectare ierarhică, elaborați schema scăzătorului elementar.

❽ Aplicînd metoda de proiectare ierarhică, elaborați schema unui **scăzător** de n biți.

❾ Cîte porți logice *NU*, *ŞI*, *SAU* va conține un scăzător de 16 biți? Dar de 32 de biți?

❿ 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 sînt prezentate în *figura 5.13*.

Sumatorul este un circuit combinațional destinat adunării a două numere binare. Tabelul de adevăr şi schema sumatorului au fost studiate în paragraful precedent.

Comparatorul este un circuit combinațional care compară numerele binare A şi B, indicînd prin cele trei ieşiri una dintre situațiile posibile: $A < B$, $A > B$ sau $A = B$.

Codificatorul este un circuit combinațional care efectuează conversiunea mesajelor s_1, s_2, ..., s_n în cuvintele binare din codul respectiv. Se consideră că fiecare dintre mesajele s_i este reprezentat prin valorile $x_1 = 0$, ..., $x_i = 1$, ..., $x_n = 0$ aplicate la intrarea codificatorului, iar cuvîntul de cod – prin variabilele de ieşire y_1, y_2, ..., y_m.

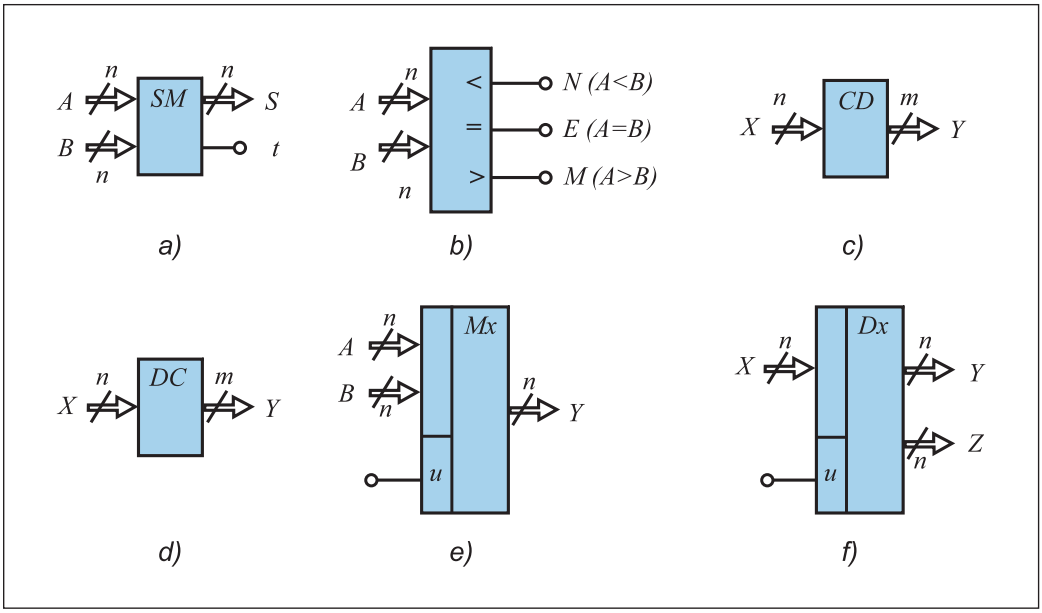


Fig. 5.13. Circuite combinaționale frecvent utilizate:
a – sumatorul; *b* – comparatorul; *c* – codificatorul; *d* – decodificatorul;
e – multiplexorul; *f* – demultiplexorul

De exemplu, variabilele x_1, x_2, x_3, ... pot reprezenta starea tastelor <*A*>, <*B*>, <*C*>, ... ale tastaturii. Codificatorul respectiv va furniza la ieşire cuvîntul de cod *ASCII* corespunzător tastei acționate.

Decodificatorul este un circuit combinațional care generează semnalul logic 1 pe o ieşire distinctă pentru fiecare combinație a valorilor variabilelor de intrare. Cu alte cuvinte, decodificatorul efectuează operația inversă a unui codificator.

Decodificatoarele sînt utilizate pentru a determina operațiile pe care trebuie să le execute procesorul, pentru selectarea unităților de intrare-ieşire, sintetizarea simbolurilor etc.

Multiplexorul este un circuit combinațional destinat selectării fluxurilor de date. În *figura 5.13e* este prezentat un multiplexor care transmite la ieşire 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 ieşirile *Y* ($u = 0$) sau *Z* ($u = 1$). Drept exemplu, amintim transferul informației de la o singură sursă la mai mulți destinatari.

Întrebări şi exerciții

❶ Explicați destinația circuitelor combinaționale frecvent utilizate: sumatorul, comparatorul, codificatorul, decodificatorul, multiplexorul şi demultiplexorul.

❷ Alcătuiți tabelul de adevăr al unui comparator de 2 biți.

❸ Cîte intrări şi cîte ieşiri poate avea un codificator? Cîte intrări şi cîte ieşiri poate avea un decodificator?

❹ Pe panoul de comandă al imprimantei sînt montate butoanele *ON LINE* (funcţionare sub controlul unităţii centrale), *OFF LINE* (funcţionare autonomă), *LINE FEED* (avans de linie) şi *FORM FEED* (avans de pagină). Alcătuiţi tabelul de adevăr al codificatorului care furnizează la ieşire următoarele combinaţii binare:

00 – *ON LINE;*

01 – *OFF LINE;*

10 – *LINE FEED;*

11 – *FORM FEED*.

❺ Pe panoul de comandă al imprimantei sînt montate becurile (diodele luminiscente) indicatoare *READY* (disponibilă), *PAPER* (lipsa de hîrtie), *TEST* (regimul de testare) şi *LOAD* (regimul de încărcare a informaţiei). Alcătuiţi tabelul de adevăr al decodificatorului care aprinde becurile în cauză. Stările respective sînt codificate prin următoarele combinaţii binare:

00 – *READY;*

01 – *PAPER;*

10 – *TEST;*

11 – *LOAD*.

❻ Tastatura calculatorului include tastele funcţionale <*F1*>, <*F2*>, ..., <*F12*>. Alcătuiţi tabelul de adevăr al codificatorului care va furniza la ieşire numărul binar corespunzător tastei funcţionale acţionate.

❼ Unităţile de intrare-ieşire 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*.

Alcătuiţi tabelul de adevăr al decodificatorului care va selecta unitatea indicată de adresa respectivă.

❽ Operaţiile aritmetice ale unui calculator ipotetic sînt codificate după cum urmează:

adunarea	– 000;
scăderea	– 001;
înmulțirea	– 010;
împărțirea	– 011;
compararea	– 100.

Alcătuiți tabelul de adevăr al decodificatorului operațiilor aritmetice.

❾ Alcătuiți 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 ieşire depind nu numai de combinațiile valorilor variabilelor de intrare, dar şi 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 sînt alcătuite din circuite combinaționale şi 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***.

Circuitul are două intrări notate cu R şi S şi două ieşiri notate cu Q şi $\overline{Q}$. Se observă că semnalele de ieşire Q şi $\overline{Q}$ sînt aplicate la intrările porților *SAU-NU*. Conexiunile respective sînt numite **reacții**. Tocmai datorită acestor conexiuni, circuitul dat are două stări distincte şi, prin urmare, asigură memorarea unui bit de informație.

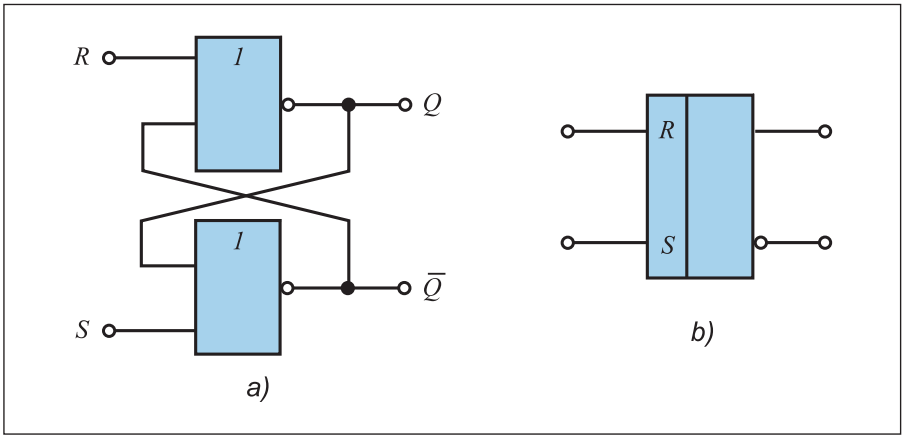


Fig. 5.14. Schema bistabilului asincron *RS* (*a*) şi simbolul utilizat (*b*)

Într-adevăr, admitem că semnalele de intrare $R = S = 0$, iar ieşirile $Q = 1$, $\bar{Q} = 0$. Datorită reacţiei, valoarea $Q = 1$ impune ieşirea $\bar{Q} = 0$. La rîndul ei, tot datorită reacţiei, valoarea $\bar{Q} = 0$ confirmă ieşirea $\bar{Q} = 1$. Exact la fel, în cazul în care ieşirile $Q = 0$, $\bar{Q} = 1$, datorită reacţiei, valoarea $\bar{Q} = 1$ impune ieşirea $Q = 0$. Prin urmare, valorile de intrare $R = S = 0$ nu schimbă starea bistabilului, acesta păstrînd cifra binară memorată.

S-a convenit ca stării $Q = 1$, $\bar{Q} = 0$ să-i corespundă cifra binară 1, iar stării $Q = 0$, $\bar{Q} = 1$ cifra binară 0. Ieşirea Q este numită ieşire directă sau ieşire adevărată, iar ieşirea $\bar{Q}$ ieşire inversă sau ieşire negată. Starea bistabilului este indicată de ieşirea directă Q.

Să examinăm cazul în care $R = 0$, $S = 1$. Presupunem că bistabilul se află în starea 0, adică $Q = 0$ şi $\bar{Q} = 1$. În acest caz, valoarea $S = 1$ va impune $\bar{Q} = 0$, care, la rîndul ei, prin reacţie, va asigura $Q = 1$. Prin urmare, bistabilul trece în starea 1 ($Q = 1$, $\bar{Q} = 0$). Această stare, după cum s-a stabilit anterior, se va păstra şi 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 ($Q = 1$, $\bar{Q} = 0$), iar $S = 0$, şi $R = 1$, bistabilul va trece în starea 0 ($Q = 0$, $\bar{Q} = 1$). Această stare se va păstra şi 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**.

Combinaţia de intrare $R = 1$ şi $S = 1$ impune valorile de ieşire $Q = 0$ şi $\bar{Q} = 0$. Valorilor de ieşire $Q = \bar{Q} = 0$ nu le corespunde nicio stare distinctă a bistabilului. Prin urmare, pentru bistabilul RS combinaţia de intrare $R = S = 1$ este o **combinaţie interzisă**.

Regimurile de funcţionare a bistabilului în cauză sînt totalizate în *tabelul 5.1*.

Tabelul 5.1

Regimurile de funcţionare ale bistabilului asincron *RS*

Intrări		***Regim de funcţionare***	***Ieşirea Q***
R	***S***		
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 influenţă a semnalelor de comandă R şi S asupra stării bistabilului. Din analiza circuitului prezentat în *figura 5.14* rezultă că semnalele de comandă, aplicate la intrările R şi S în momente arbitrare, pot schimba starea bistabilului.

Circuitele secvenţiale în care starea circuitului poate fi schimbată de semnalele de comandă în momente arbitrare de timp se numesc circuite asincrone.

Un calculator modern conţine zeci de mii de bistabile. Schimbarea stării lor în momente arbitrare, greu de controlat, poate fi cauza unor erori de funcţionare. Pen-

tru 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 sînt 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***.

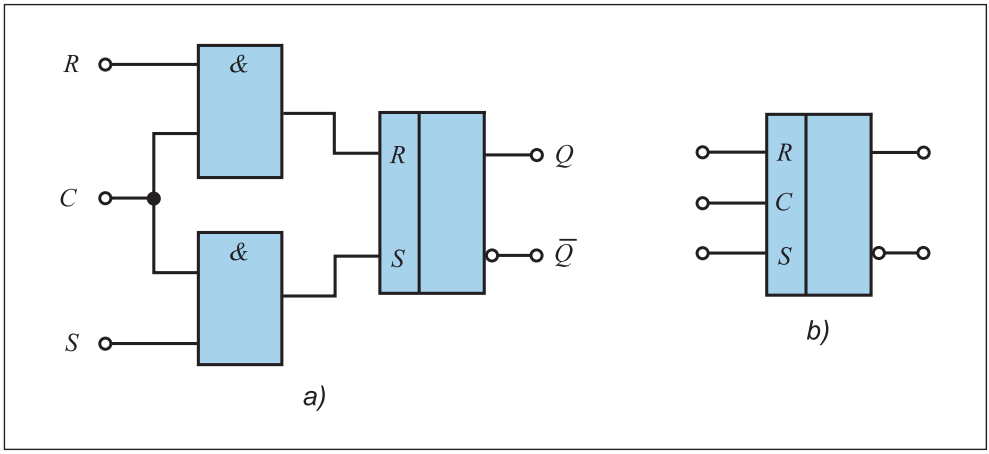


Fig. 5.15. Schema bistabilului sincron *RS* (*a*) și simbolul utilizat (*b*)

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 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?

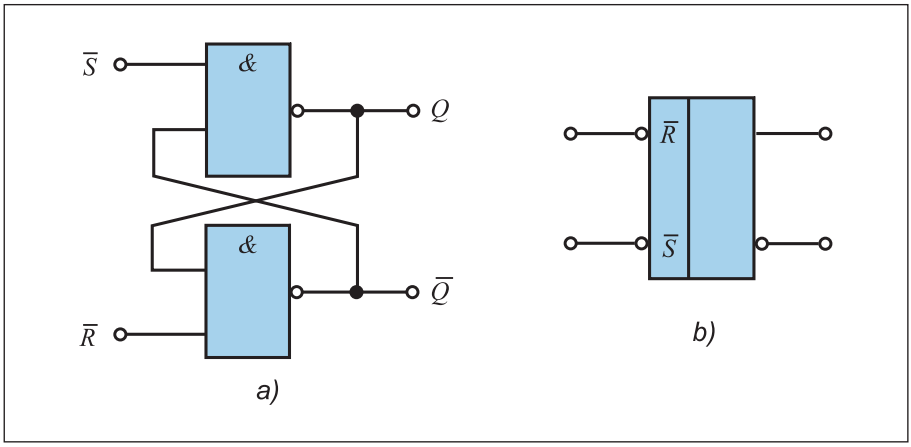


Fig. 5.16. Schema bistabilului asincron $\overline{R}\overline{S}$ (*a*) şi simbolul utilizat (*b*)

❼ În *figura 5.16* este prezentată schema circuitului bistabil de bază realizat cu porți logice *ŞI-NU*, denumit **bistabil asincron** $\overline{R}\overline{S}$.
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?

❽ Utilizînd bistabilul asincron $\overline{R}\overline{S}$, elaborați schema **bistabilului sincron** $\overline{R}\overline{S}$.

❾ Desenați schema detaliată (la nivelul porților logice *ŞI*, *SAU-NU*) a bistabilului sincron *RS*. Pentru aceasta decalcați schemele respective din *figurile 5.14a* şi *5.15a*.

❿ Desenați schema detaliată (la nivelul porților logice *ŞI, ŞI-NU*) a bistabilului sincron $\overline{R}\overline{S}$.

5.6. Circuite secvențiale frecvent utilizate

Registrul (*fig. 5.17a*) este un dispozitiv numeric destinat păstrării temporare a unui număr binar,

$$D = d_{n-1} \ldots d_1 d_0.$$

Registrul este constituit din bistabile la care sînt ataşate circuite combinaționale care 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 şi împărțirea numerelor binare, scrierea sau citirea datelor pe disc, transmiterea datelor prin firele telefonice etc., apare necesitatea deplasării în stînga sau în dreapta a informației memorate într-un registru. În acest scop se utilizează **registrele de deplasare** (*fig. 5.17 b, c*).

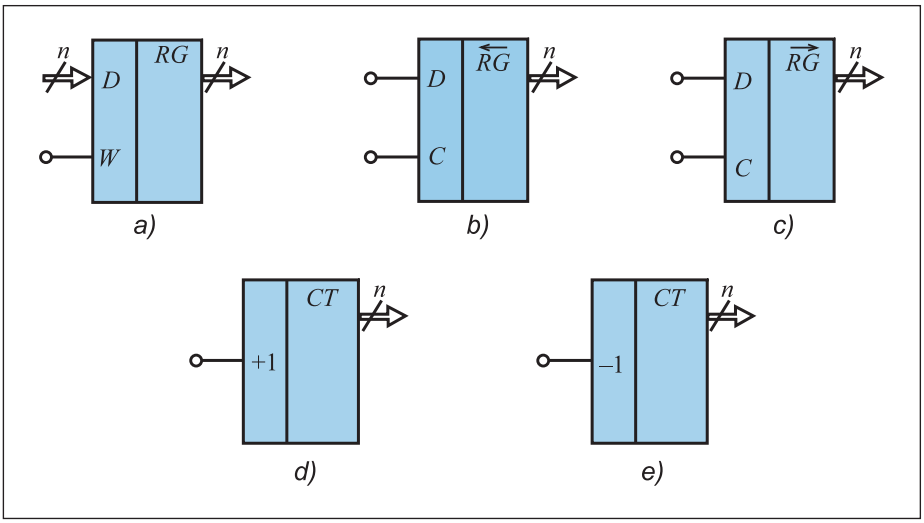


Fig. 5.17. Circuite secvențiale frecvent utilizate:
a – registrul; *b* – registru de deplasare de la dreapta spre stînga; *c* – registru de deplasare de la stînga spre dreapta; *d* – numărător direct; *e* – numărător invers

Succesiunea stărilor unui **registru de deplasare de la dreapta spre stînga** este dată în *figura 5.18*.

Timpul	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. Succesiunea stărilor unui registru de deplasare de la dreapta spre stînga

Se consideră că starea inițială a registrului este 0101, iar impulsurile sînt aplicate la intrarea C în momentele de timp t_1, t_2, t_3 etc. Fireşte, registrul în cauză calculează produsul $D \times 2$.

În mod similar, un **registru de deplasare de la stînga spre dreapta** va calcula cîtul $D : 2$.

Numărătoarele sînt 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 şi reversibile).

În general, numărătoarele sînt realizate prin asocierea circuitelor bistabile cu circuite combinaționale care determină modul corect în care numărătoarele urmează să-şi 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. Considerînd 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.

Timpul	d_2	d_1	d_0
inițial	0	0	0
t_1	0	0	1
t_2	0	1	0
t_3	0	1	1
t_4	1	0	0
t_5	1	0	1
t_6	1	1	0
t_7	1	1	1
t_8	0	0	0
t_9	0	0	1
...		...	

Fig. 5.19. Succesiunea stărilor unui numărător binar direct de trei biți

Numărătoarele care își schimbă starea conform tabelului din *figura 5.19* sînt 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 În registrul de deplasare de la dreapta spre stînga (*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 stînga spre dreapta este încărcat unul dintre următoarele cuvinte binare:

a) 00000; *f)* 00001;

b) 10000; *g)* 10001;

c) 01000; *h)* 01010;

d) 00100; *i)* 01100;

e) 00010; *j)* 00110.

Care va fi conținutul registrului după aplicarea la intrarea *C* a două impulsuri consecutive?

❹ Elaborați un program PASCAL care simulează funcționarea unui registru de deplasare de la stînga spre dreapta de *n* biți.

❺ Care este destinația numărătoarelor? Cum se schimbă stările unui numărător direct? Dar ale unui numărător invers?

❻ 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?

❼ 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.

❽ Elaborați un program PASCAL 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 întîrziere.

Elementul de întîrziere reprezintă un circuit electronic care realizează funcția logică de repetare $y = x$, însă semnalul de ieșire y repetă semnalul de intrare x cu o întîrziere 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 întîrziere. Pentru a mări „inertitatea" circuitelor electronice și pentru a realiza întîrzieri semnificative, în componența lor se includ condensatoare și rezistoare. În acest caz întîrzierea Δ este determinată de capacitatea și rezistența componentelor respective.

În *figura 5.20* sînt prezentate simbolul și diagramele în timp ale elementului de întîrziere.

Schema unui **generator de impulsuri periodice** realizat pe baza unui element de întîrziere 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 întîrziere se menține valoarea logică 1. Cînd la intrare se aplică semnalul de pornire $x = 1$, ieșirea devine egală cu 0 (*fig. 5.22*). În continuare, valoarea logică 0, după o întîrziere Δ, este aplicată la a doua intrare a porții logice *ȘI-NU*. Astfel, ieșirea y devine egală cu 1. Valoa-

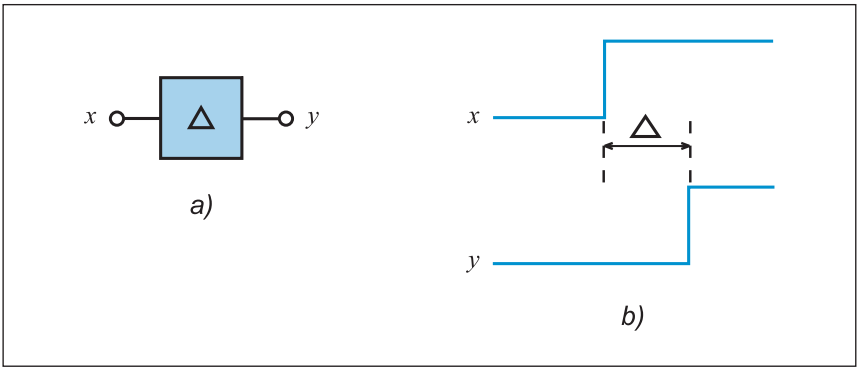


Fig. 5.20. Simbolul (*a*) şi diagramele în timp (*b*) ale elementului de întîrziere

rea logică 1, după o întîrziere Δ, va fi din nou aplicată la intrarea porţii logice *ŞI-NU*, impunînd la ieşire valoarea $y = 0$ etc.

Prin urmare, la ieşirea 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 $x = 0$.

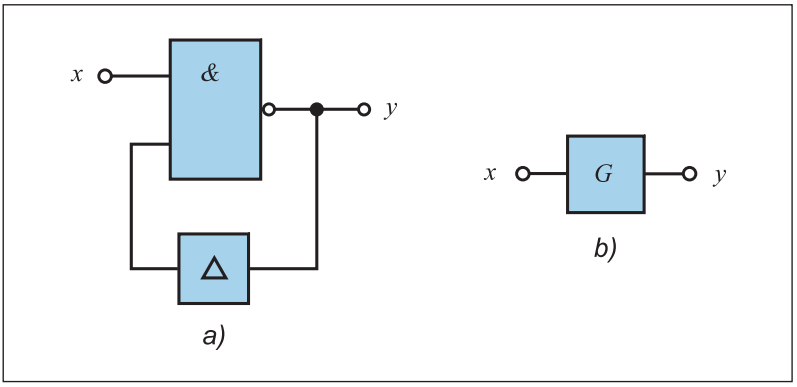


Fig. 5.21. Schema (*a*) şi simbolul generatorului de impulsuri periodice (*b*)

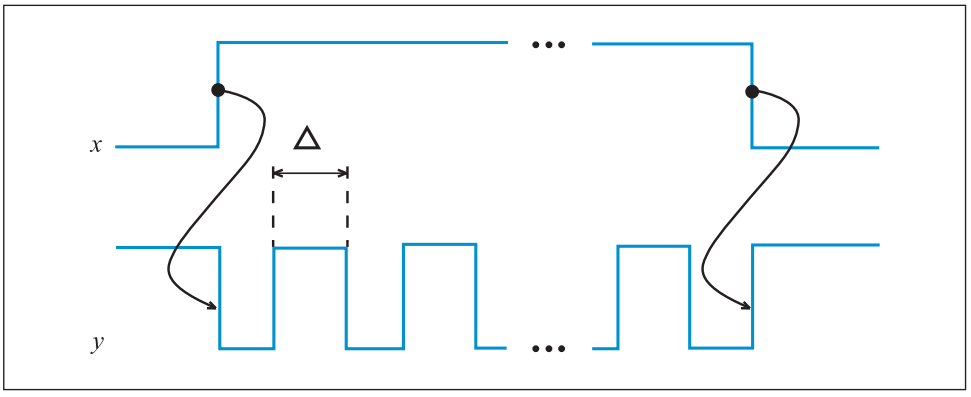


Fig. 5.22. Diagramele în timp ale generatorului de impulsuri periodice

Întrebări şi exerciţii

❶ Care este destinaţia elementului de întîrziere? Desenaţi diagramele în timp ale elementului examinat.

❷ Explicaţi cum funcţionează generatorul de impulsuri periodice. De ce depinde durata impulsurilor?

❸ Înlocuiţi poarta logică *ŞI-NU* din componenţa generatorului de impulsuri periodice prezentat în *figura 5.21* printr-o poartă logică *SAU-NU*. Explicaţi cum va funcţiona circuitul respectiv. Desenaţi diagramele în timp ale generatorului obţinut.

❹ E cunoscut faptul că variaţiile mărimilor fizice nu pot avea loc instantaneu. Prin urmare, orice poartă logică are o întîrziere δ, denumită **întîrziere parazitară**, valoarea concretă a căreia depinde de particularităţile circuitului respectiv.

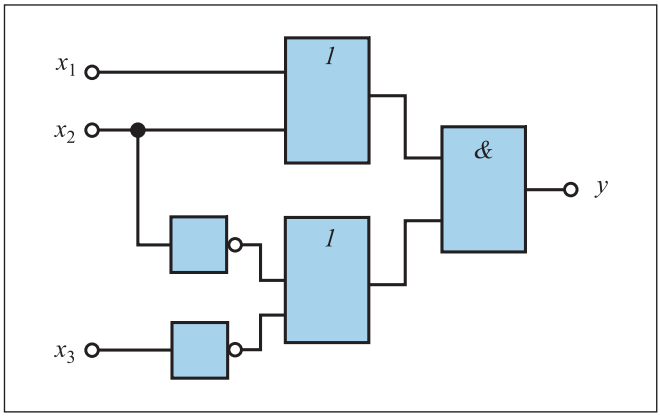
Excludeţi din schema prezentată în *figura 5.21* elementul de întîrziere, aplicînd semnalul de la ieşirea porţii logice *ŞI-NU* direct la intrarea ei. Explicaţi cum va funcţiona circuitul obţinut. De ce depinde durata impulsurilor la ieşirea porţii logice? Desenaţi diagramele respective în timp.

❺ Circuitul secvenţial constă dintr-o poartă logică *NU,* semnalul de ieşire al căreia este aplicat direct la intrarea ei. Cum va funcţiona acest circuit?

Test de autoevaluare nr. 5

1. Desenaţi schema circuitului logic pentru calcularea funcţiei $y = \bar{x}_1 x_2 \vee x_1 x_3$.

2. La intrările circuitului logic de mai jos sînt aplicate valorile $x_1 = 1$, $x_2 = 0$ şi $x_3 = 1$. Desenaţi schema în caiet şi indicaţi pe desen valorile semnalelor de la intrările şi ieşirile fiecărei porţi logice.



3. Scrieţi funcţia logică realizată de circuitul din itemul 2.

4. Elaboraţi un program PASCAL care simulează funcţionarea semisumatorului. Cifrele binare *a* şi *b* se citesc de la tastatură, iar cifra-sumă *s* şi cifra de transport *t* se afişează pe ecran.

5. Cîte porți logice *NU*, *ŞI*, *SAU* va conține un sumator de 8 biți? Argumentați răspunsul.

6. Elaborați un program PASCAL care simulează funcționarea sumatorului de 8 biți. Numerele binare *A* şi *B* se citesc de la tastatură, iar suma *S* şi cifra de depăşire *t* se afişează pe ecran.

7. Indicați corespondența între denumirile circuitelor combinaționale frecvent utilizate (coloana din stînga) şi destinația acestora (coloana din dreapta):

	(a) efectuează conversiunea mesajelor în cuvinte binare;
(1) sumatorul;	(b) efectuează conversiunea semnalelor;
(2) comparatorul;	(c) distribuie fluxurile de date;
(3) codificatorul;	(d) efectuează conversiunea cuvintelor binare în mesaje;
(4) decodificatorul;	(e) calculează suma a două numere binare;
(5) multiplexorul;	(f) selectează obiectele unei imagini;
(6) demultiplexorul.	(g) compară două numere binare;
	(h) selectează fluxurile de date.

8. Pe panoul de comandă al unui joc electronic sînt montate becurile (diodele luminiscente) indicatoare SUS, JOS, STÎNGA şi DREAPTA. Alcătuiți tabelul de adevăr al decodificatorului care aprinde becurile în cauză. Comenzile respective sînt codificate prin următoarele combinații binare:

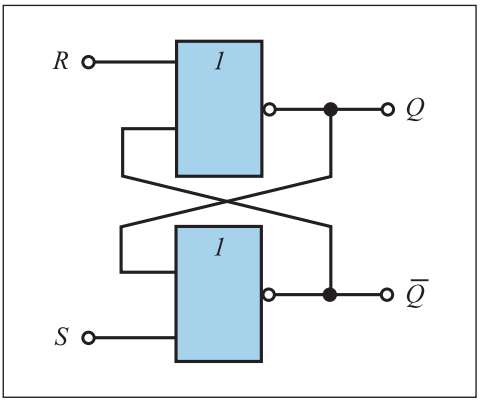
00 – SUS;

01 – JOS;

10 – STÎNGA;

11 – DREAPTA.

9. Cunoscînd valorile variabilelor de intrare $R = 0$ şi $S = 1$, determinați valorile funcțiilor Q şi $\overline{Q}$ la ieşirea bistabilului de mai jos.



10. Desenați schema detaliată (la nivelul porților logice *ŞI, ŞI-NU*) a bistabilului sincron $\overline{R}\overline{S}$.

11. Indicați corespondența între denumirile circuitelor secvențiale frecvent utilizate (coloana din stînga) şi destinația acestora (coloana din dreapta):

(1)	bistabilul;	(a)	deplasează cuvîntul binar de la stînga la dreapta;
(2)	registrul;	(b)	transformă cuvintele binare în caractere;
(3)	registrul de deplasare de la dreapta spre stînga;	(c)	adună cîte o unitate la conținutul său;
(4)	registrul de deplasare de la stînga spre dreapta;	(d)	memorează o cifră binară;
(5)	numărătorul direct;	(e)	adună două numere binare, reprezentate în cod invers;
(6)	numărătorul invers.	(f)	scade cîte o unitate din conținutul său;
		(g)	deplasează cuvîntul binar de la dreapta la stînga;
		(h)	scade numerele binare, reprezentate în cod direct;
		(i)	memorează un cuvînt binar.

12. În registrul de deplasare de la stînga spre dreapta este încărcat cuvîntul binar 10011011. Care va fi conținutul registrului după aplicarea la intrarea *C* a trei impulsuri consecutive?

13. În registrul de deplasare de la dreapta spre stînga este încărcat cuvîntul binar 11011101. Care va fi conținutul registrului după aplicarea la intrarea *C* a patru impulsuri consecutive?

14. Elaborați un program PASCAL care simulează funcționarea unui registru de deplasare de la dreapta spre stînga. Conținutul inițial al registrului *R* şi numărul de impulsuri de intrare *m* se citesc de la tastatură, iar rezultatul deplasării se afişează pe ecran.

15. Un numărător direct de 8 biți se află în starea inițială 10001101. Care va fi starea numărătorului după aplicarea a 6 impulsuri de intrare?

16. Un numărător invers de 8 biți se află în starea inițială 10101110. Care va fi starea numărătorului după aplicarea a 5 impulsuri de intrare?

17. Elaborați un program PASCAL care simulează funcționarea unui numărător direct. Conținutul inițial al numărătorului *A* şi numărul de impulsuri de intrare *m* se citesc de la tastatură, iar conținutul final al numărătorului se afişează pe ecran.

Capitolul 6

STRUCTURA ŞI FUNCȚIONAREA CALCULATORULUI

6.1. Schema funcțională a calculatorului

Schema funcțională a calculatorului numeric este prezentată în *figura 6.1*.

Conform acestei scheme, calculatorul numeric conține următoarele **unități funcționale**:

– o unitate de memorie pentru a înmagazina datele inițiale, intermediare şi finale ale problemei, precum şi instrucțiunile care indică secvența calculelor;

– un dispozitiv aritmetic şi logic necesar efectuării operațiilor aritmetice şi logice elementare;

– unul sau mai multe dispozitive de intrare, respectiv, ieşire, necesare comunicării din exterior cu calculatorul;

– un dispozitiv central de comandă şi control care generează o succesiune de semnale de comandă necesare executării secvențiale a instrucțiunilor.

Dispozitivul aritmetic şi logic şi dispozitivul central de comandă formează **unitatea centrală de prelucrare a informației** sau, mai pe scurt, **procesorul**.

Memoria calculatoarelor moderne este organizată în două niveluri, şi anume: o unitate de memorie internă cu o viteză mare de lucru şi una sau mai multe unități de memorie externă cu o viteză mai redusă, însă cu o capacitate mult mai mare decît cea a memoriei interne.

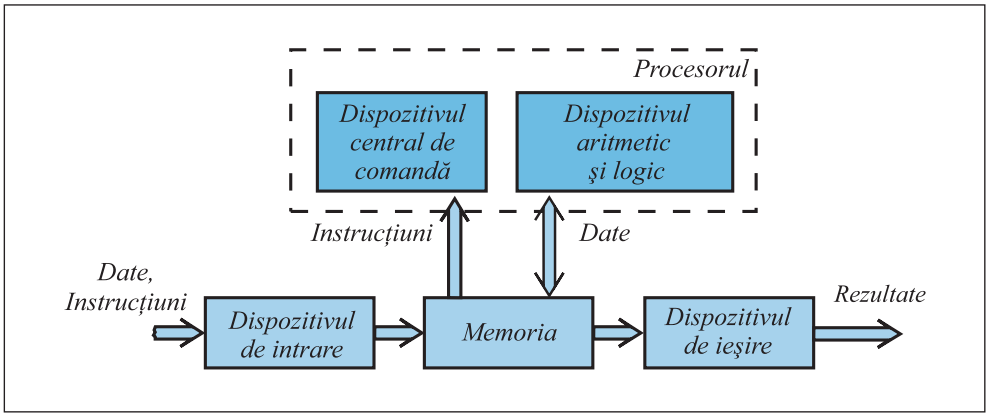


Fig. 6.1. Schema funcțională a calculatorului

Memoria internă (numită uneori memorie principală, centrală sau operativă) păstrează programul în curs de executare şi datele folosite de acesta, prezenţa ei fiind o condiţie esenţială pentru funcţionarea calculatorului.

Memoria externă are rolul de a păstra cantităţi mari de informaţie şi programe folosite frecvent pentru a putea fi aduse într-un interval de timp mic în memoria internă. În prezent ca memorii externe sînt utilizate unităţile cu discuri sau benzi magnetice, unităţile cu discuri optice etc.

Unităţile de memorie externă şi dispozitivele de intrare-ieşire sînt numite **echipamente periferice**.

Pentru a asigura o interacţiune eficientă a procesorului, a memoriei interne şi a echipamentelor periferice, în cazul calculatoarelor personale schema funcţională se realizează fizic conform schemei-bloc prezentate în *figura 6.2*.

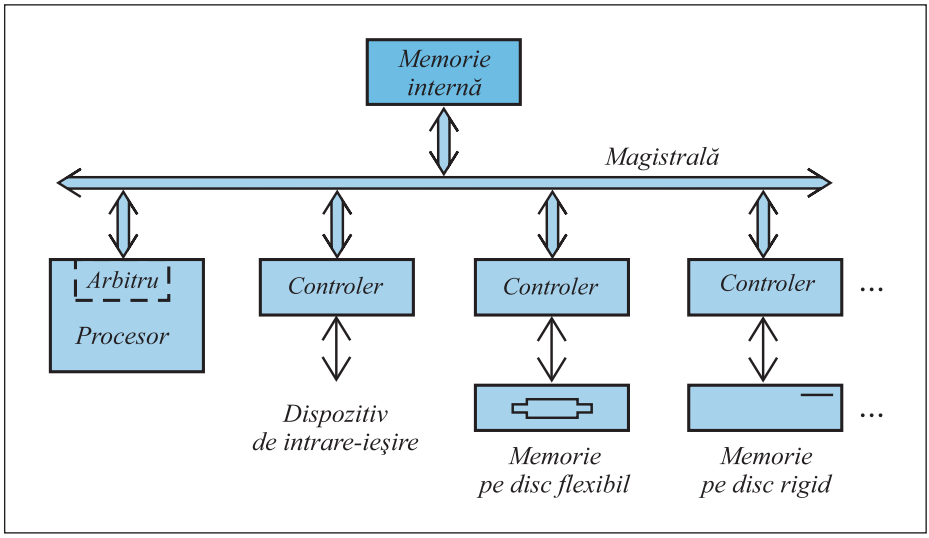


Fig. 6.2. Schema-bloc a unui calculator personal

Din analiza *figurii 6.2* rezultă că toate calculatoarele moderne au o **configuraţie modulară**. Fiecare modúl, şi anume controlerul, imprimantele, unităţile de disc magnetic etc., funcţionează şi, în consecinţă, pot fi incluse sau excluse din componenţa calculatorului independent unul de altul. Prin urmare, configuraţia calculatorului poate fi modificată în funcţie de destinaţia 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 şi 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 şi vizualizatoare de o rezoluţie adecvată, cu tastaturi similare pupitrului regizoral etc.

Întrebări şi exerciţii

❶ Numiţi unităţile funcţionale ale calculatorului şi explicaţi destinaţia lor.

❷ Care este rolul memoriei interne? Cum se realizează memoria externă a calculatoarelor moderne?

❸ Numiţi echipamentele periferice pe care le cunoaşteţi dvs.

❹ Numiţi componentele unui calculator personal şi explicaţi destinaţia lor.

❺ Care este structura şi cum interacţionează componentele unui calculator?

❻ Cum poate fi modificată configuraţia unui sistem de calcul? Care sînt avantajele configuraţiei modulare a calculatorului?

❼ Desenaţi schema-bloc a calculatorului cu care lucraţi dvs. Care componente sînt obligatorii şi care opţionale pentru funcţionarea calculatorului?

❽ 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?

6.2. Formatul instrucţiunilor

Pentru a rezolva o problemă, calculatorul trebuie să cunoască în fiecare moment atît operaţia pe care urmează să o execute, cît şi datele care participă în operaţie. Aceste operaţii sînt comunicate calculatorului prin intermediul instrucţiunilor.

Instrucţiunea calculatorului reprezintă o succesiune de cifre binare prin care se indică procesorului operaţia de executat şi amplasamentul (locul) operanzilor.

Succesiunea binară respectivă, denumită uneori şi **cuvînt instrucţiune**, este împărţită în cîmpuri, fiecare cîmp avînd o semnificaţie precisă. Numărul şi semnificaţia cîmpurilor poartă denumirea de **formatul instrucţiunii**. În *figura 6.3* sînt prezentate formatele utilizate în calculatoarele moderne.

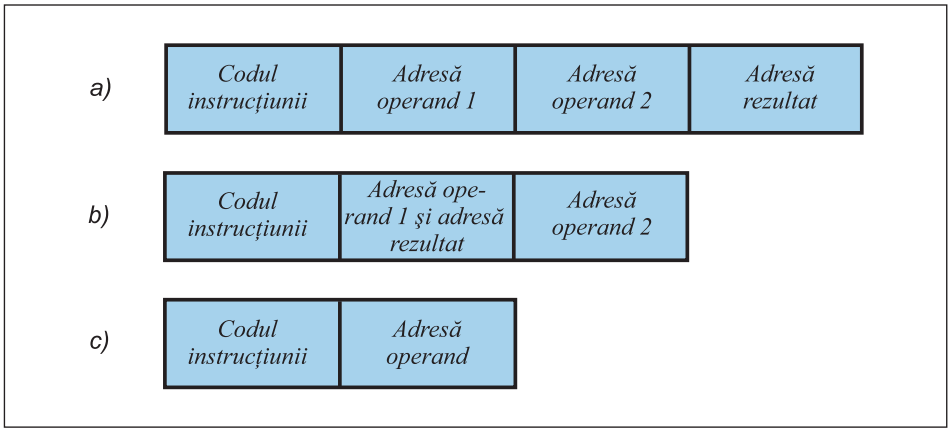


Fig. 6.3. Formatul instrucţiunilor cu trei (*a*), două (*b*) şi o singură adresă (*c*)

În general, pentru executarea unei operații este necesar ca instrucțiunea să conțină trei adrese (*fig. 6.3a*). Primele două adrese sînt folosite pentru obținerea celor doi operanzi asupra cărora se va efectua operația specificată de cîmpul *Codul instrucțiunii*. Rezultatul operației va fi depus pe adresa specificată de cîmpul *Adresă rezultat*.

Să analizăm un exemplu. Presupunem că operațiile aritmetice şi logice sînt codificate după cum urmează (pentru simplitate vom utiliza echivalentele zecimale ale cîmpurilor binare respective):

01 – adunarea;
02 – scăderea;
03 – operația logică *ŞI*;
04 – operația logică *SAU*.

Instrucțiunea

```
01 100 110 215
```

va impune procesorul să adune numerele din locațiile 100 şi 110 şi 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 *ŞI* asupra biților cuvintelor din locațiile 200 şi 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 şi aceluiaşi program pentru prelucrarea oricăror date inițiale. Faptul că instrucțiunile lucrează cu adrese al căror conținut trebuie prelucrat, şi nu cu însuşi conținutul, constituie un principiu fundamental al calculatoarelor numerice, care permite ca un program să fie elaborat şi introdus în calculator independent de datele concrete asupra cărora se aplică.

În formatul cu trei adrese (*fig. 6.3a*), adresele sînt **specificate explicit**. Pentru o reprezentare mai compactă a instrucțiunilor se utilizează **specificarea implicită** a unor adrese. În acest caz, cuvîntul instrucțiune nu conține un cîmp 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 şi 110 şi 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 răspîndit, permite scrierea de programe avînd un număr de instrucțiuni comparabil cu cel obținut atunci cînd s-ar utiliza mai multe adrese.

Formatul cu o singură adresă (*fig. 6.3c*) se utilizează în calculatoarele procesorul cărora include un registru special, denumit **acumulator**. În acumulator se păstrează primul operand şi se depune rezultatul executării operației respective. Prin urmare, adresa primului operand şi adresa rezultatului sînt 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 sînt eficiente din punctul de vedere al lungimii cuvîntului şi al rapidității calculatorului. Totuşi un program scris cu instrucțiuni avînd o singură adresă va conține mai multe instrucțiuni decît 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 cîmpul *Codul instrucțiunii*.

Întrebări şi exerciții

❶ Enumerați formatele instrucțiunilor utilizate în calculatoarele moderne. Explicați modul de specificare implicită a unor adrese.

❷ Explicați semnificația cîmpurilor instrucțiunilor cu trei sau două adrese.

❸ Cum se specifică adresele operanzilor şi 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 şi logice sînt codificate ca şi în exemplul din paragraful de față.

❺ Care va fi conținutul locației 100 după executarea instrucțiunii

```
01 200 300 100,
```

dacă în locațiile 200 şi 300 sînt înscrise numerele 17 şi, respectiv, 31?

❻ 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.`

❼ Care va fi conținutul locației 200 după executarea instrucțiunii

```
01 200 100,
```

dacă în locațiile 100 şi 200 sînt înscrise numerele 18 şi, respectiv, 32?

❽ Explicați cum vor fi executate următoarele instrucțiuni cu o singură adresă:

a) 01 100; *c)* 02 400;

b) 03 200; *d)* 04 150.

❾ Care va fi conținutul acumulatorului după executarea instrucțiunii

01 100,

dacă inițial în locația 100 era înscris numărul 12, iar în acumulator numărul 26?

❿ Care sînt avantajele şi dezavantajele formatelor cu trei, două sau cu o singură adresă?

6.3. Tipuri de instrucțiuni

Instrucțiunile unui calculator se împart în patru grupe:

– instrucțiuni operaționale, care efectuează operații aritmetice şi logice asupra datelor specificate prin operanzi;

– instrucțiuni de transfer, care deplasează informația între registre şi/sau locații fără a modifica informația transferată;

– instrucțiuni de salt, care în urma verificării unor condiții, modifică analiza şi execuția secvențială a instrucțiunilor din program;

– instrucțiuni de intrare-ieşire care permit comunicarea calculatorului cu exteriorul.

Instrucțiunile operaționale prelucrează datele păstrate în locațiile memoriei interne şi în registrele procesorului. Cele mai cunoscute instrucțiuni ale acestei grupe sînt cele care efectuează operațiile aritmetice de bază: *adunarea*, *scăderea*, *înmulțirea* şi *împărțirea*.

Instrucțiunile logice de tipul *ŞI*, *SAU*, *NU* sînt instrucțiuni operaționale care acționează asupra pozițiilor individuale ale informației binare. În categoria instrucțiunilor operaționale întîlnim şi instrucțiuni de tipul: *şterge* conținutul unei locații sau al unui registru, *complementează* conținutul unei locații, *creşte* cu o unitate conținutul unui registru etc. În fine, în categoria instrucțiunilor operaționale sînt incluse instrucțiunile de *deplasare* a informației, în care partea de adresă a instrucțiunii conține un număr întreg, care specifică numărul pozițiilor cu care se face deplasarea.

Instrucțiunile de transfer deplasează informația dintre locațiile memoriei interne, între registre sau între locații şi registre fără a altera conținutul informației transferate. Instrucțiunea trebuie să specifice explicit sau implicit adresa-sursă şi adresa de destinație a transferului. În timpul transferului şi după transfer, informația din sursă rămîne neschimbată. Cele mai uzuale instrucțiuni ale acestei grupe sînt cele prin care conținutul unei locații trece într-un anumit registru, registrul acumulator, precum şi 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 sînt analizate şi execu-

tate în mod secvențial, în ordinea în care sînt aşezate 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-ieşire permit comunicarea calculatorului cu echipamentele periferice. Echipamentul cu care se va efectua operația de intrare-ieşire se specifică în partea de adresă a instrucțiunii. De regulă, instrucțiunile de acest tip conțin atît informații legate de natura schimbului de date, adică introducerea sau extragerea lor, cît şi comenzi necesare funcționării corecte a periferiei. Tot în aceste instrucțiuni se specifică şi registrele sau locațiile în care vor fi depuse sau din care vor fi luate datele respective.

Întrebări şi exerciții

❶ Cum se clasifică instrucțiunile unui calculator? Care este destinația instrucțiunilor din fiecare grupă?

❷ Dați cîteva exemple de instrucțiuni operaționale. Estimați numărul instrucțiunilor operaționale posibile.

❸ Care este destinația instrucțiunilor de transfer? Estimați numărul instrucțiunilor posibile de transfer.

❹ Cînd şi cum se utilizează instrucțiunile de salt? Care condiții de test pot fi analizate de aceste instrucțiuni?

❺ Care este destinația instrucțiunilor de intrare-ieşire? Ce informații conțin aceste instrucțiuni?

6.4. Limbajul cod calculator şi limbajul de asamblare

Pentru a rezolva o problemă, în memoria calculatorului trebuie să fie încărcate programul respectiv şi datele de prelucrat. Instrucțiunile programului şi datele de prelucrat se înmagazinează în memoria internă sub forma unor succesiuni de cifre binare pe care dispozitivul central de comandă le poate extrage şi interpreta.

> ***Programele reprezentate în formă de succesiuni binare direct executabile de calculator se numesc programe în limbaj cod calculator sau programe în limbaj maşină.***

Pentru utilizator programul în cod calculator poate fi prezentat în formă de şiruri 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 şi ineficient. Pentru a simplifica procesul de elaborare a programelor, s-a convenit ca instrucţiunile să fie scrise într-un limbaj simbolic, denumit **limbaj de asamblare**. În acest limbaj codurile instrucţiunilor se reprezintă printr-un grup de caractere, astfel ales încît să sugereze cît mai bine natura operaţiei. Acest grup de caractere, de regulă trei, este cunoscut sub numele **mnemonica instrucţiunii**.

De exemplu, codurile instrucţiunilor calculatorului ipotetic din paragraful precedent pot fi notate simbolic conform *tabelului 6.1*.

Tabelul 6.1

Mnemonica instrucţiunilor

Cod instrucţiune	*Mnemonica*	*Semnificaţia*
01	INC	*Încarcă acumulatorul*
02	MEM	*Memorează acumulatorul*
03	ADU	*Adunare*
04	SCD	*Scădere*
05	SLT	*Salt necondiţionat*
06	SLTC	*Salt condiţionat*
07	STP	*Stop*

Adresele locaţiilor memoriei interne pot fi specificate prin denumiri simbolice alese de utilizator, denumiri care sugerează semnificaţia conţinutului locaţiilor respective.

De exemplu, locaţia 185, în care se va păstra numărul *x,* poate fi notată prin *X*, locaţia 213 pentru numărul *y* se va nota prin *Y*, locaţia 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* şi *y* va avea forma:

```
INC X
ADU Y
MEM S.
```

În general, există o corespondenţă directă între scrierea instrucţiunii în limbajul de asamblare şi scrierea în limbajul cod calculator, ceea ce face uşoară translatarea (traducerea) limbajului de asamblare în limbajul cod calculator.

Translatarea constă în înlocuirea mnemonicii instrucţiunii şi a adreselor simbolice prin şirurile binare respective. Această înlocuire este făcută de un program special, denumit program de asamblare sau asamblor.

Limbajele cod calculator şi de asamblare sînt **limbaje dependente de calculator**. Această dependenţă constă în faptul că formatul, codurile şi mnemonica instrucţiunilor exprimă structura internă a calculatorului. Programele elaborate în aceste limbaje sînt cele mai scurte şi 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 operaţiile de prelucrare şi tipurile de date nu sînt legate de echipamentele calculatorului. Însă, cu regret, detaşarea utilizatorului de structura internă a calculatorului diminuează eficacitatea programelor respective.

Întrebări şi exerciţii

❶ Care este diferenţa dintre limbajul cod calculator şi limbajul de asamblare?

❷ Cum se exprimă codurile instrucţiunilor şi adresele locaţiilor într-un limbaj de asamblare?

❸ Care este destinaţia şi cum se realizează translatarea programelor scrise într-un limbaj de asamblare?

❹ Se consideră că denumirea simbolică *X* semnifică locaţia 100, denumirea *Y* – locaţia 101, iar denumirea *S* – locaţia 102. Exprimaţi în limbajul de asamblare (vezi *tabelul 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 conţinutul locaţiilor 100, 101 şi 102 pînă şi după execuţia fiecărui program?

❺ Se consideră că denumirile simbolice *X*, *Y* şi *S* specifică, respectiv, locaţiile 100, 200 şi 300. Translataţi 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
```

Explicaţi cum va fi executat fiecare program.

❻ Care este diferenţa dintre limbajele dependente şi limbajele independente de calculator? Care sînt avantajele şi dezavantajele fiecărui limbaj?

6.5. Resursele tehnice şi resursele programate ale calculatorului

Numărul total al instrucţiunilor unui calculator depinde, în primul rînd, de capacitatea lui. În cazul unui sistem mare de calcul acest număr poate depăşi 1000, pe cînd la calculatoarele foarte mici el nu este mai mare decît 100. O anumită operaţie poate fi efectuată la unele calculatoare cu o singură instrucţiune, pe cînd aceeaşi operaţie, în alte calculatoare, pentru care nu există o instrucţiune specifică, este efectuată cu ajutorul unei succesiuni de alte instrucţiuni existente.

Operaţiile efectuate intern de către componentele electronice ale calculatorului sînt cunoscute ca **operaţii implementate prin echipamente**, pe cînd operaţiile efectuate cu ajutorul unei secvenţe de instrucţiuni sînt cunoscute ca **operaţii implementate prin program**. De exemplu, operaţia de extragere a rădăcinii pătrate într-un anumit calculator poate fi efectuată prin echipament electronic, în alt calculator — printr-un subprogram. Delimitarea dintre implementarea prin echipamente şi program depinde de calculator. În *figura 6.4* este prezentată o astfel de delimitare în cazul unui calculator cu posibilităţi medii.

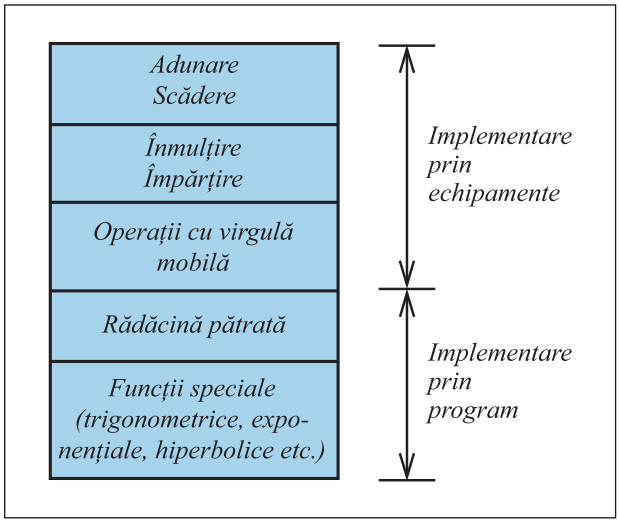


Fig. 6.4. Delimitarea între implementarea prin echipamente şi prin program

Operaţiile implementate prin echipamente se efectuează prin execuţia unei singure instrucţiuni, pe cînd o operaţie implementată prin program necesită execuţia mai multor instrucţiuni. Prin urmare, operaţiile implementate prin echipamente se efectuează mai rapid, însă calculatorul respectiv este mai complex şi, în consecinţă, mai scump. Din contra, operaţiile implementate prin program se efectuează mai lent, însă calculatoarele respective sînt mai simple şi, evident, mai ieftine.

Din analiza principiilor de funcţionare a procesorului rezultă că toate echipamentele unui calculator devin inutile în absenţa programelor care guvernează efectuarea operaţiilor necesare pentru rezolvarea unei probleme. Exact în acelaşi mod, programele sînt fără niciun folos în absenţa echipamentelor numerice res-

pective. Prin urmare, utilizarea tehnicii de calcul este posibilă numai în prezența atît a echipamentelor, denumite **resurse tehnice**, cît şi 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-ieşire etc. **Resursele programate** vor include subprogramele care realizează operațiile implementate prin program, programele care simplifică accesul la unitățile de intrare-ieşire, asambloarele, editoarele de texte, compilatoarele limbajelor algoritmice şi, evident, programele elaborate de fiecare utilizator.

Menționăm că în literatura de specialitate resursele tehnice uneori sînt denumite prin cuvîntul englez *hardware* („produse de metal"), iar resursele programate – prin cuvîntul *software* („produse moi"). Respectiv, implementarea prin echipamente se numeşte implementare prin *hardware*, iar implementare prin program – implementare prin *software*.

Întrebări şi exerciții

❶ De ce depinde numărul total al instrucțiunilor unui calculator?

❷ Cum se efectuează operațiile în cazurile implementării prin echipamente şi în cele ale implementării prin program?

❸ Analizînd repertoriul instrucțiunilor calculatorului la care lucrați dvs., determinați cum sînt implementate următoarele operații:
– înmulțirea şi împărțirea numerelor binare;
– adunarea şi scăderea numerelor cu virgulă mobilă;
– înmulțirea şi împărțirea numerelor cu virgulă mobilă;
– extragerea rădăcinii pătrate;
– calcularea funcțiilor trigonometrice.

❹ Care sînt avantajele şi dezavantajele implementării prin echipamente? Dar ale implementării prin program?

❺ Numiți resursele tehnice şi resursele programate ale unui sistem de calcul. Care sînt resursele respective în cazul calculatorului la care lucrați dvs.?

6.6. Memorii externe pe benzi şi discuri magnetice

Principiul de funcționare a memoriilor examinate constă în înregistrarea informației pe un strat magnetic, aflat în mişcare. 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 foloseşte 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*.

Capul constă dintr-un miez, de regulă din tole (foițe) de permalloy foarte subțiri (0,05 mm) şi o înfăşurare.

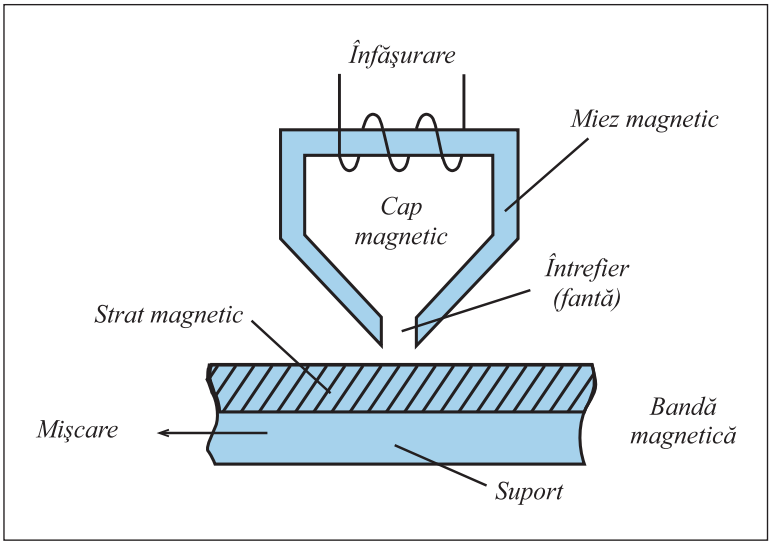


Fig. 6.5. Cap magnetic pentru înscrierea şi citirea informaţiei

Într-un strat magnetic neînregistrat, cîmpurile magnetice ale particulelor de oxid de fier sînt orientate haotic, anulîndu-se reciproc. Pentru **a înregistra** cifra binară 0 sau 1, prin înfăşurarea capului magnetic se trece impulsul respectiv de curent. Impulsul care traversează înfăşurarea creează în întrefier un cîmp magnetic intens care magnetizează stratul aflat în momentul actual sub cap. Direcţia de magnetizare, deci informaţia binară înscrisă, depinde de direcţia curentului în înfăşurarea capului magnetic. În *figura 6.6* este prezentat un exemplu de înregistrare a informaţiei binare 101101 pe un strat magnetic aflat în mişcare.

Distanţa b determină lungimea porţiunii necesare pentru a memora o cifră binară. Valoarea lui b depinde de viteza de mişcare a suportului, de proprietăţile fizice ale stratului magnetic, de elementele constructive ale capului magnetic etc.

Numărul de elemente de memorie binară pe unitatea de lungime a suportului se numeşte densitatea de înregistrare a informaţiei.

În cazul înregistrărilor magnetice densitatea este dată de mărimea $1/b$. Valorile practice variază după tipul dispozitivului de memorare şi firma constructoare, fiind de ordinul sutelor şi miilor de biţi pe milimetru de lungime a suportului.

În timpul operaţiei de **citire** cîmpul magnetic al particulelor de oxid de fier, trecînd prin dreptul întrefierului (*fig. 6.5*), induce în înfăşurarea capului magnetic un semnal de ordinul 10^{-3} volţi. Acest semnal este amplificat şi 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 informaţia spre/de la memoria internă după receptarea comenzilor corespunzătoare de la procesorul calculatorului. O unitate de memorie pe bandă magnetică (*fig. 6.7*) include mecanismul de antrenare a benzii, dispozitivul de scriere-citire şi circuitele de comandă aferente.

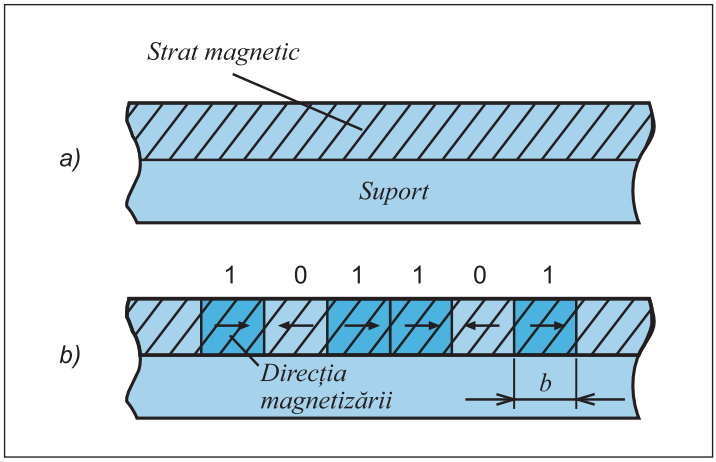


Fig. 6.6. Starea stratului magnetic pînă (*a*) şi după înregistrare (*b*)

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 sînt denumite unități de memorie externă cu **acces secvențial**.

Timpul necesar pentru selectarea unei informații din multitudinea datelor memorate pe un suport se numeşte timp de acces.

Timpul de acces al unității de bandă magnetică depinde de viteza benzii şi de locul amplasării informației de citit: la începutul, la sfîrşitul sau la mijlocul benzii. În cazul informațiilor înregistrate la sfîrşitul benzii, timpul de acces este de ordinul minutelor.

Capacitatea de memorare a unei benzi magnetice depinde de densitatea înregistrării, numărul de piste, lungimea benzii şi are valori de ordinul 10^8 octeți.

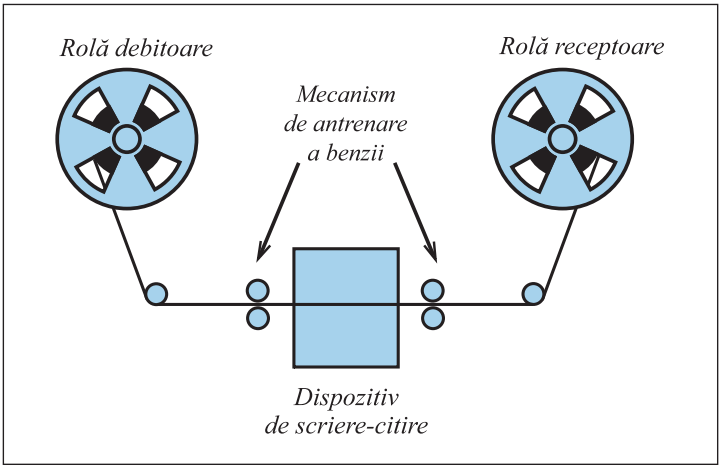


Fig. 6.7. Unitate de bandă magnetică

Din cauza timpului mare de acces şi capacitatea de memorie relativ mică, benzile magnetice se utilizează, în general, numai pentru arhivarea informaţiei.

Unitatea de discuri magnetice reprezintă la etapa actuală cea mai răspîndită memorie externă a calculatoarelor numerice. Suportul informaţiei este format dintr-un pachet de discuri, care poate fi fix sau amovibil, rotit cu o viteză de ordinul miilor de turaţii pe minut. Discurile sînt acoperite cu un strat de material feromagnetic. Informaţia este înregistrată pe ambele feţe de-a lungul unor piste concentrice. Pentru aceasta fiecare faţă este explorată în plan orizontal de către un cap magnetic mobil (*fig. 6.8*).

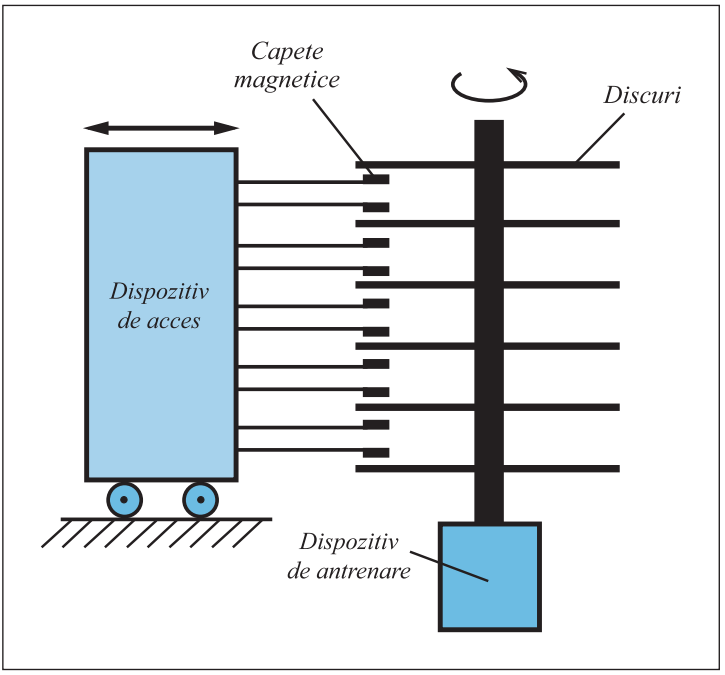


Fig. 6.8. Unitate de discuri cu capete mobile

Capetele sînt montate pe un braţ extensibil şi acţionate de un mecanism pentru a ajunge în dreptul pistei selectate.Toate capetele unei unităţi de memorie cu discuri sînt poziţionate simultan.

Timpul de acces al unităţii de discuri se compune din timpul necesar deplasării ansamblului de capete magnetice de pe cilindrul curent pe cilindrul indicat şi din timpul necesar ca sectorul respectiv al discului să ajungă în dreptul capului magnetic. În practică se utilizează **timpul mediu de acces**, care pentru unităţile de disc moderne este de ordinul 10^{-3} secunde.

Amintim că în calculatoarele de performanţă se utilizează unităţi de discuri cu capete magnetice fixe – cîte un cap magnetic pentru fiecare pistă. Aceste unităţi asigură un timp de acces de ordinul 10^{-4} secunde, însă sînt foarte scumpe.

Pentru schimbul de informaţii între calculatoare se utilizează discuri singulare din material plastic, denumite **discuri flexibile** sau **dischete**. Aceste discuri sînt încorporate într-o casetă din plastic sau în plicuri speciale. Organizarea fizică a

informației este aceeași ca și în cazul pachetelor de discuri, însă unitățile respective sînt mult mai simple și, evident, mai ieftine. Pentru a le deosebi de discurile flexibile, discurile convenționale ale calculatoarelor personale sînt 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.

Întrebări și exerciții

❶ Cum sînt reprezentate cifrele binare 0 și 1 în înregistrările magnetice?
❷ Care este destinația capului magnetic?
❸ De ce depinde densitatea de înregistrare magnetică a informației?
❹ Cum se citește informația înregistrată pe un strat magnetic?
❺ Explicați cum funcționează unitatea de memorie pe bandă magnetică din *figura 6.7*.
❻ De ce depinde capacitatea de memorare a unei benzi magnetice?
❼ O bandă magnetică are lungimea de 750 de metri. Înregistrarea informației se efectuează pe 8 piste plus pista bitului de paritate. Capacitatea de memorare a benzii este 47 *Megaocteți*. Determinați densitatea de înregistrare.
❽ Viteza benzii magnetice este 2 m/s. În rola debitoare (*fig. 6.7*) sînt 750 m de bandă. Determinați timpul de acces la informația de la mijlocul benzii.
❾ Cum funcționează o unitate de discuri cu capete mobile?
❿ Cum este organizată informația pe un pachet de discuri magnetice?
⓫ Care este diferența dintre memoriile externe cu acces direct și acces secvențial?
⓬ De ce depinde timpul de acces al unității de discuri magnetice?
⓭ Determinați capacitatea discului flexibil cu care lucrați dvs.
⓮ Pentru unitatea de disc rigid cu care lucrați dvs. determinați:
– capacitatea discului;
– timpul mediu de acces.

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* (***C**ompact **D**isc – **R**ead **O**nly **M**emory*).

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* (***C**ompact **D**isc – **R**ecordable*).

3) Discuri optice **reinscriptibile**. Discurile în cauză permit mai multe cicluri de scriere/ştergere a informaţiei. Abrevierea acestor discuri este *CD-RW* (***C**ompact **D**isc – **R**e**W**ritable*).

Pentru a asigura compatibilitatea unităţilor de citire, formatul datelor şi dimensiunile discurilor optice sînt standardizate. În *figura 6.9* este reprezentată structura discului optic *numai pentru citire* destinat publicului larg.

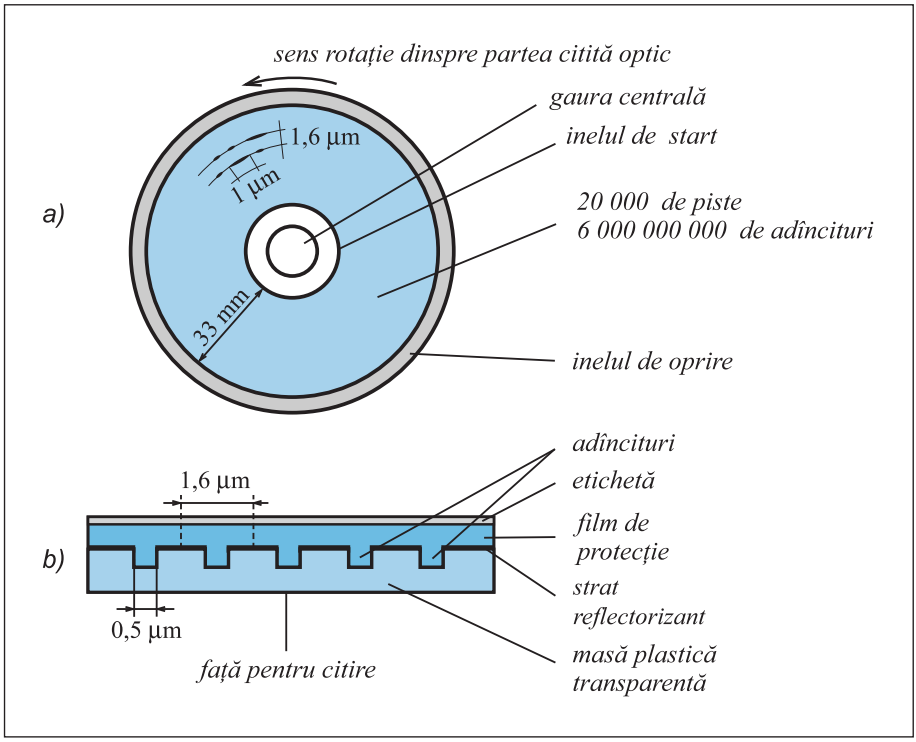


Fig. 6.9. Structura discului optic *CD-ROM*: *a* – poziționarea pistelor pe disc; *b* – secțiune a discului perpendiculară pe piste

Înregistrarea cifrelor binare pe astfel de discuri constă dintr-o succesiune de adîncituri (în limba engleză *pit*) realizate pe una dintre suprafeţele discului. Aceste adîncituri sînt despărţite de mici pauze şi sînt plasate pe suprafaţa discului sub forma unei piste în spirală.

Dimensiunile adîncîturilor sînt de ordinul unui micron (1 micron = 10^{-3} milimetri), distanţa dintre spire este de 1,6 microni, lungimea spiralei fiind de 5 300 de metri. Discul conţine 20 000 de piste (spire) pe care se află circa $6 \cdot 10^9$ adîncituri. Capacitatea de memorare a discului este de 640 *Megaocteţi*.

Citirea discului optic se realizează cu ajutorul unui fascicul de lumină care, după ce se reflectă de suprafaţa activă, este interceptat de o celulă fotosensibilă (*fig. 6.10*).

Parcurgînd pistele respective, fasciculul laser este reflectat cînd stratul reflectorizant se află în **punctul de focalizare** şi nereflectat în caz contrar. Cu alte cuvinte, adîncîturile de pe suprafaţa activă a discului optic schimbă (modulează) intensitatea fasciculului reflectat. În consecinţă, la ieşirea celulei fotosensibile se

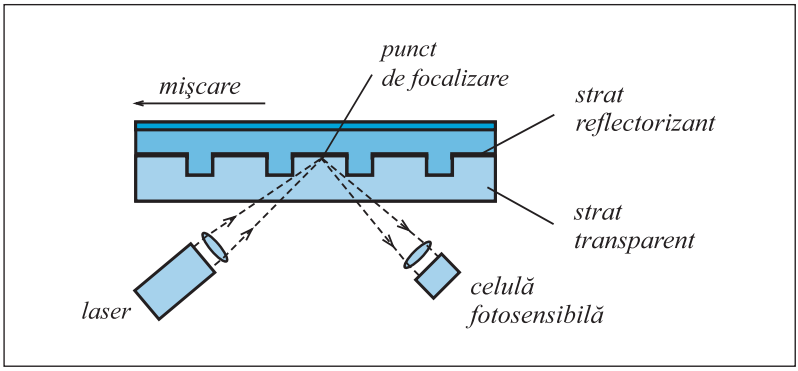


Fig. 6.10. Citirea discurilor optice

formează un semnal care redă succesiunea de cifre binare 0, 1 înregistrată pe disc la etapa fabricării.

În unităţile de disc optic moderne sursa de lumină – laserul şi celula fotosensibilă – fotodioda se realizează într-un dispozitiv integral, denumit **cap optic de citire**. Viteza unghiulară a discului este de 200–600 de turaţii pe minut, iar viteza liniară este de 1,4 m/s. În pofida vitezelor foarte mari, discul optic practic nu se uzează, întrucît între capul optic şi disc nu există un contact mecanic direct. Prin urmare, durata de exploatare a unui disc *CD-ROM* este determinată de calitatea stratului reflectorizant şi a stratului de protecţie. În cazul utilizării aluminiului, din cauza oxidării, stratul reflectorizant se întunecă, reducînd astfel viaţa unui disc optic la 10–15 ani. În cazul unui strat reflectorizant din aur numai o acţiune 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, mediilor de programare, enciclopediilor, jocurilor electronice şi a altor informaţii destinate unui număr foarte mare de utilizatori.

Structura **discurilor optice inscriptibile** şi **reinscriptibile** este redată în *figura 6.11*.

Din figură se observă că discurile examinate conţin 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.

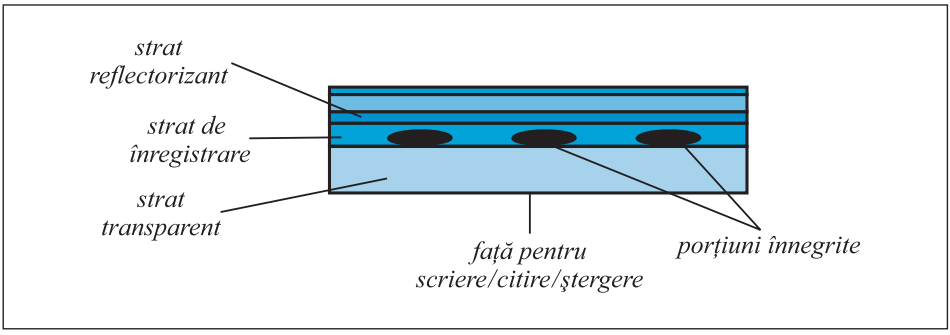


Fig. 6.11. Structura discurilor optice inscriptibile şi reinscriptibile

Înregistrarea informației se realizează cu ajutorul unui fascicul laser foarte puternic care înnegreşte porțiunile respective ale stratului de înregistrare.

La citire, porțiunile întunecate blochează trecerea fasciculului laser spre stratul reflectorizant, modificînd astfel intensitatea luminii captate de celula fotosensibilă (*fig. 6.10*). Întrucît 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 pînă la o temperatură specială, denumită **temperatură critică**. Evident, după ştergere, pe discul reinscriptibil pot fi înscrise date binare noi. Discurile moderne *CD-RW* suportă pînă la 10 000 de cicluri de scriere/ştergere.

Neajunsul comun al discurilor inscriptibile şi reinscriptibile este faptul că ele sînt mult mai sensibile la temperatură, durata lor de exploatare fiind mai mică decît a discurilor *CD-ROM*. Aceste discuri sînt şi mai scumpe, întrucît pentru o citire sigură stratul reflectorizant este realizat din aur sau argint.

De obicei, discurile optice *CD-R* şi *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 şapte ori mai mare decît a unui disc de tipul *CD*.

Întrebări şi exerciții

❶ Cum sînt reprezentate cifrele binare 0, 1 în înregistrările optice?

❷ Care este capacitatea de memorare a unui disc optic?

❸ Cum se înregistrează informația pe un disc optic *CD-ROM*?

❹ De ce depinde capacitatea de memorare a unui disc optic?

❺ 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.

❻ Care este domeniul de utilizare a discurilor *CD-ROM*?

❼ De ce depinde timpul de acces la informațiile de pe un disc optic?

❽ Viteza liniară a discului optic este de 1,4 m/s. Cunoscînd 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.

❾ Pe cele 20 000 de piste (spire) ale unui disc optic sînt înregistrați circa 640 de *Megaocteți*. Cîtă informație conține o singură pistă a discului optic?

❿ Pe un disc *CD-ROM* sînt înregistrate în formă binară circa 74 de minute de muzică. Determinați cîtă informație va conține un cîntec cu durata de 4 minute şi 30 de secunde. Cîte piste va ocupa cîntecul respectiv?

⓫ Cum se citeşte informaţia de pe un disc optic? Care este destinaţia capului optic de citire?

⓬ Explicaţi destinaţia straturilor unui disc optic inscriptibil. Cum se realizează înregistrarea informaţiei pe discul în cauză?

⓭ De ce depinde durata de exploatare a unui disc *CD-ROM* ? Dar a discurilor *CD-R* şi *CD-RW* ?

⓮ Cum se înregistrează şi se şterge informaţia de pe un disc optic reinscriptibil?

⓯ Care este domeniul de utilizare a discurilor *CD-R* şi *CD-RW* ?

⓰ Aflaţi parametrii tehnici ai unităţii de disc optic cu care este dotat calculatorul dvs. Memorizaţi regulile de exploatare a discurilor şi a unităţilor de memorie externă pe discuri optice.

6.8. Vizualizatorul şi tastatura

Vizualizatorul este un dispozitiv de ieşire prin intermediul căruia informaţia este prezentată utilizatorului pe ecranul unui tub catodic. Schema funcţională a vizualizatorului este prezentată în *figura 6.12*.

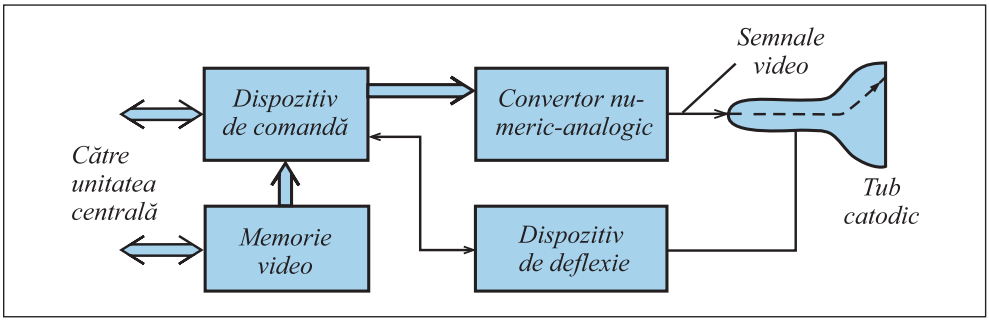


Fig. 6.12. Schema funcţională a vizualizatorului

Ca şi în cazul televizoarelor pentru publicul larg, imaginea pe ecranul tubului catodic se formează din puncte. Punctele sînt activate de un fascicul de electroni. Culoarea şi luminanţa 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 stînga la dreapta şi de sus în jos, este asigurată de dispozitivul de deflexie.

Fiecărui punct al ecranului îi corespunde o locaţie în memoria video a vizualizatorului. Locaţiile conţin informaţii referitoare la culoarea şi luminanţa punctelor respective. Dispozitivul de comandă citeşte locaţiile memoriei video în ordinea parcurgerii punctelor de pe ecran. Conţinutul fiecărei locaţii este interpretat drept o instrucţiune referitoare la modul de activare a punctului care îi corespunde. Semnalele video, necesare pentru comanda fasciculului de electroni, sînt formate de convertoare numeric-analogice.

Memoria video a vizualizatorului este încărcată de unitatea centrală a calculatorului. În orice moment, calculatorul poate modifica conţinutul acestei memo-

rii. În consecință, se va schimba şi imaginea afişată pe ecran. Imaginile pot fi animate prin modificarea conținutului memoriei video la frecvențe utilizate în cinematografie.

Vizualizatorul poate funcţiona 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 afişat un singur caracter dintr-un set de 256 de caractere. Setul de caractere este constituit din literele mari şi mici ale alfabetului latin, cifrele zecimale, simbolurile matematice, semnele de punctuație şi unele caractere semigrafice, utilizate la afişarea pe ecran a tabelelor, diagramelor, chenarelor etc. Fiecare zonă-caracter poate avea o culoare pentru caracter şi altă culoare pentru fundal, ceea ce îi permite utilizatorului să afişeze pe ecran texte cu litere de diferite culori.

În regim grafic utilizatorul poate controla afişarea pe ecran a fiecărui punct. Numărul de puncte pe orizontală şi verticală determină rezoluția vizualizatorului. De exemplu, expresia „rezoluția 640×200" înseamnă că vizualizatorul are 640 de puncte pe orizontală şi 200 pe verticală.

Există mai multe norme internaționale care reglementează rezoluția şi numărul de culori ale vizualizatoarelor. Aceste caracteristici sînt recunoscute şi respectate de firmele producătoare.

De exemplu, în cazul calculatoarelor personale cele mai răspîndite sînt normele denumite *EGA*, *VGA* şi *SVGA*. Norma *EGA* stipulează că vizualizatorul are o rezoluție de 640×350 de puncte, permițînd reprezentarea a 64 de culori.

Norma *VGA*, păstrînd compatibilitatea cu *EGA*, oferă ca facilități suplimentare rezoluția 640×480 de puncte şi 256 de culori distincte.

În scopul îmbunătățirii calității imaginii redate, norma *SVGA* adaugă rezoluția suplimentară de 1024×768 de puncte.

Menționăm că realizările tehnologice din ultimii ani au permis înlocuirea tuburilor catodice cu ecrane plate, fapt ce asigură o reducere substanțială a dimensiunilor vizualizatorului şi o îmbunătățire semnificativă a rezoluției imaginilor afişate. Astfel de vizualizatoare se caracterizează printr-un consum redus de energie şi rezoluții de 1920×1200 de puncte.

Tastatura este un dispozitiv de intrare care transformă acționarea unei taste într-un cuvînt 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 ieşire se furnizează cuvintele unui cod binar, de obicei standard (*ISO*, *ACSII* etc.). Unele tipuri de tastaturi pot fi prevăzute cu un generator audio, care la acționarea tastelor produce un sunet specific.

La noi, ca şi în țările anglofone, cea mai răspîndită este tastatura de tipul *QWERTY*, denumirea căreia provine de la amplasarea caracterelor *Q*, *W*, *E*, *R*, *T* şi *Y* în rîndul de sus al tastelor alfanumerice. În țările francofone se utilizează tastatura *AZERTY*, în Germania *QWERTZ* etc.

Deşi locul amplasării tastelor şi numărul lor pot să difere, destinația tastelor este aceeaşi.

Tastatura dispune de următoarele grupe de taste: alfanumerice, funcţionale şi speciale. Grupul de taste alfanumerice include tastele cifrelor zecimale, tastele caracterelor alfabetului englez, tastele simbolurilor matematice şi ale semnelor de punctuaţie. Grupul de taste funcţionale include tastele <*F1*>, <*F2*>, ... ,<*F12*>. Tastele respective nu au o destinaţie prestabilită şi semnificaţia lor este definită de programul care derulează pe calculator. Tastele speciale se utilizează pentru poziţionarea cursorului vizualizatorului, pentru introducerea în calculator a unor cuvinte binare care nu au taste proprii etc.

În calculatoarele de performanţă, vizualizatorul şi tastatura pot forma un echipament periferic unitar, denumit **consolă**. Consola utilizată de operator pentru dirijarea proceselor de calcul se numeşte **monitor**.

Întrebări şi exerciţii

❶ Explicaţi cum funcţionează vizualizatorul. Care este destinaţia părţilor componente ale unui vizualizator?

❷ Cum poate fi schimbată imaginea afişată pe ecranul unui vizualizator? Cum pot fi animate imaginile de pe ecran?

❸ Prin ce se deosebesc regimurile de funcţionare a vizualizatorului?

❹ Care sînt indicii principali de calitate ai unui vizualizator?

❺ Determinaţi tipul vizualizatorului la care lucraţi dvs. Aflaţi rezoluţia şi numărul de culori disponibile.

❻ Care sînt părţile componente ale unei tastaturi? Cum se determină tipul tastaturii?

❼ Numiţi grupurile de taste şi destinaţia lor.

6.9. Imprimantele

Imprimantele sînt dispozitive de ieşire care furnizează rezultatele sub forma unui document tipărit. În funcţie de **tehnica de tipărire** utilizată, imprimantele se pot clasifica în:

– imprimante mecanice, în care tipărirea se face prin acţionarea unor ciocănaşe sau ace;

– imprimantele laser, în care tipărirea se face folosind metode electrostatice ca la maşinile de copiat;

– imprimantele cu jet de cerneală;

– imprimantele termice, funcţionarea cărora se bazează pe utilizarea unei hîrtii speciale care îşi schimbă culoarea în funcţie de temperatură.

Principiul de funcţionare a **imprimantei matriciale cu ace** este prezentat în *figura 6.13*.

Capul de imprimare conţine un grup de ace metalice subţiri care la momentul potrivit lovesc prin bandă tuşată în foaia de hîrtie. Configuraţia ácelor ce lovesc în fiecare moment şi avansarea capului de-a lungul liniei de imprimat determină imaginile tipărite.

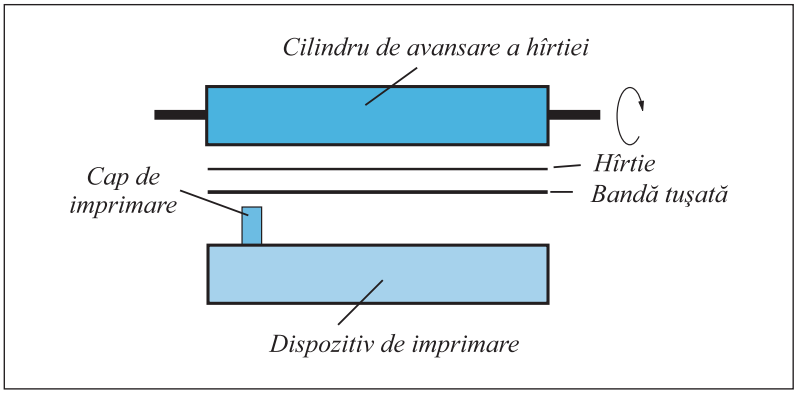


Fig. 6.13. Principiul de funcționare a imprimantei matriciale cu ace

Imprimantele matriciale pot funcţiona în regim grafic şi regim alfanumeric.

În **regim grafic,** calculatorul comandă tipărirea pe hîrtie a fiecărui punct aparte. Evident, din puncte pot fi formate orice imagini: grafice, desene tehnice, caractere cu configuraţii concepute de utilizator. Imprimarea informaţiei în regim grafic este lentă din cauza deplasărilor multiple ale capului de imprimare şi a caracterului discontinuu al avansului de hîrtie cu paşi 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 sînt 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 caractere, imaginile cărora au fost, în prealabil, înmagazinate în memoria imprimantei. Imprimantele matriciale simple au cîteva seturi de caractere, înscrise într-o memorie permanentă. Imprimantele matriciale mai performante permit încărcarea prin program a mai multe seturi de caractere cu configuraţii concepute de utilizator.

E cazul să subliniem că cu cît numărul ácelor din capul de imprimare este mai mare, cu atît calitatea imprimării este mai bună. În prezent se utilizează imprimante cu 9 sau cu 24 de ace. Calitatea imprimării poate fi îmbunătăţită prin tipărirea repetată (de 2-4 ori) a aceluiaşi caracter pe acelaşi loc. Viteza de tipărire a imprimantelor mecanice cu ace este de 150–500 de caractere pe minut.

Principiul de funcţionare a **imprimantelor laser** este prezentat în *figura 6.14.*

Elementul principal al acestei imprimante este un tambur acoperit cu un strat semiconductor care îşi poate schimba proprietăţile electrice sub acţiunea luminii. Mai întîi, suprafaţa tamburului se electrizează. Cu ajutorul razelor laser pe suprafaţa electrizată a tamburului se proiectează punctele imaginii de tipărit. Întrucît sectoarele care au fost luminate îşi schimbă conductibilitatea, sarcinile electrice respective se neutralizează. Prin urmare, pe suprafaţa 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 hîrtie şi se fixează prin topirea particulelor de pulbere.

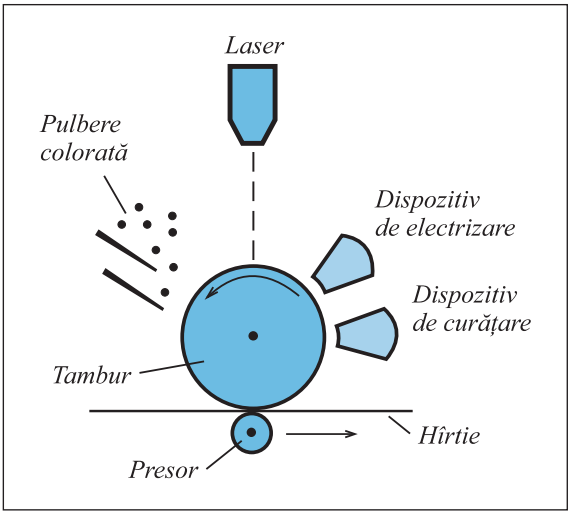


Fig. 6.14. Principiul de funcționare a imprimantei laser

În continuare, toate sarcinile electrice de pe suprafața tamburului sînt neutralizate, iar resturile de pulbere sînt curățate.

Imprimantele laser sînt cele mai bune dintre imprimantele moderne. Textele şi 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 hîrtie de nişte ajutaje speciale. Ele asigură o calitate foarte bună a tiparului şi sînt utilizate pentru imprimarea color. Aceste imprimante sînt mai scumpe decît imprimantele cu ace şi 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 rînduri pe minut, însă avantajul principal constă în dimensiunile reduse ale imprimantei.

Întrebări şi exerciții

❶ Cum se clasifică imprimantele în funcție de tehnica de tipărire?

❷ Care sînt părțile componente ale unei imprimante?

❸ Care este principiul de funcționare a imprimantei matriciale cu ace? Cum funcționează această imprimantă?

❹ Explicați cum funcționează o imprimantă laser. Care este avantajul principal al acestei imprimante?

❺ Determinați tipul imprimantei cu care lucrați dvs. Aflați parametrii tehnici ai imprimantei: setul de caractere, regimurile de funcționare, capacitatea memoriei-tampon, viteza de imprimare.

6.10. Clasificarea calculatoarelor

Caracteristica generală a unui calculator include următoarele date:
– viteza de operare;
– capacitatea memoriei interne;
– componența, capacitatea şi timpul de acces ale unităților de memorie externă;
– componența şi parametrii tehnici respectivi ai echipamentelor periferice;
– parametrii de masă şi 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 depăşeşte 20 de milioane de dolari. Cercetări şi proiectări în industria supercalculatoarelor se realizează în SUA şi Japonia de firmele *IBM*, *Gray Research*, *Fujitsu ETA Systems*, *Sutherland* etc. Supercalculatoare se utilizează în prelucrări extrem de complexe ale datelor în aeronautică, fizica nucleară, astronautică, seismologie, prognoza meteo etc.

Calculatoarele mari (în engleză *mainframe* „dulapul principal") pot executa sute de bilioane de operații pe secundă (1 bilion = 10^{12}), prețul variind între 20 de mii şi cîteva 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 şi imprimante, sute de console aflate la diferite distanțe de unitatea centrală. Aceste calculatoare se utilizează în cadrul unor mari centre de calcul şi funcționează în regim non-stop. Principalele firme producătoare de calculatoare mari sînt *IBM*, *Hitachi, Amdahl, Fujitsu* etc.

Minicalculatoarele efectuau sute de milioane de operații pe secundă, iar prețul lor nu depăşea 200–300 mii de dolari. Echipamentele periferice ale unui minicalculator includeau cîteva discuri magnetice, una sau două imprimante, mai multe console. Minicalculatoarele erau mai uşor de utilizat şi operat decît calculatoarele mari şi se utilizau în proiectarea asistată de calculator, în automatizări industriale, pentru prelucrarea datelor în experimentele ştiințifice 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 şi calculatoare personale, sînt realizate la prețuri scăzute – între 100 şi 15 000 de dolari şi 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 o unitate de disc rigid, o unitate de disc flexibil, o unitate de disc optic, o imprimantă şi o consolă. Structura modulară şi gruparea tuturor echipamentelor în jurul unei magistrale permite configurarea microcalculatorului în funcție de necesitățile individuale ale fiecărui utilizator.

Corporații care produc microcalculatoare există în foarte multe țări, însă lideri mondiali, unanim recunoscuți, sînt firmele *IBM*, *Apple*, *Hewlett-Packard*, *Dell* etc.

Întrebări şi exerciţii

❶ Numiţi parametrii mai semnificativi ai unui calculator. Determinaţi parametrii respectivi ai calculatorului cu care lucraţi dvs.

❷ Cum se clasifică calculatoarele în funcţie de parametrii tehnici şi economici?

❸ Daţi o caracteristică succintă a fiecărei categorii de calculatoare: supercalculatoare, calculatoare mari, minicalculatoare şi microcalculatoare.

❹ Folosind un server de căutare, găsiţi în Internet informaţii detaliate despre principalii producători de calculatoare personale.

6.11. Microprocesorul

Microprocesorul este un circuit integrat care implementează funcţiile unităţii centrale de prelucrare a informaţiei, şi anume – extragerea şi executarea instrucţiunilor.

De regulă, un microprocesor conţine un dispozitiv aritmetic şi altul de comandă, un grup de registre, destinate păstrării temporale a datelor frecvent utilizate, magistralele şi circuitele de comandă aferente (*fig. 6.15*).

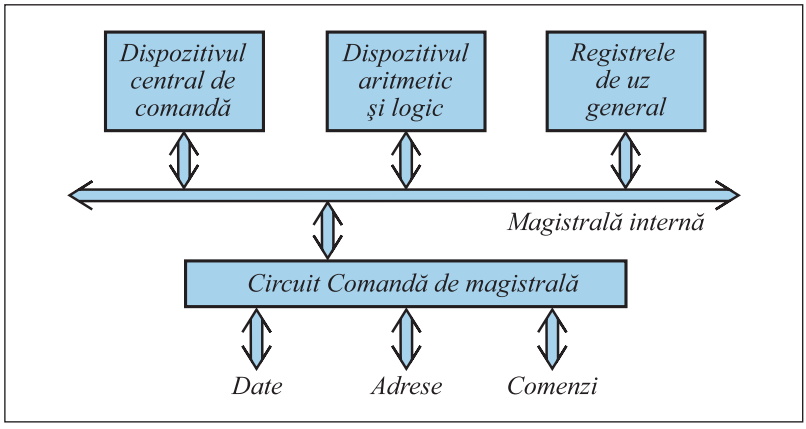


Fig. 6.15. Schema funcţională a unui microprocesor

Microprocesorul interacţionează cu unităţile de memorie şi echipamentele periferice prin intermediul a trei magistrale: *Date*, *Adrese* şi *Comenzi*. Traficul de informaţii prin magistrale este controlat de circuitul *Comandă de magistrală*.

Extragerea şi executarea instrucţiunilor are loc sub controlul **dispozitivului central de comandă**. Pentru aceasta, prin magistrala de adrese se indică adresa locaţiei 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 cuvîntului;

– frecvenţa ceasului de sistem;

– capacitatea magistralelor.

Lungimea cuvîntului reprezintă numărul de biţi ai succesiunilor binare care pot fi memorizate în registre şi prelucrate de dispozitivul aritmetic al microprocesorului. Microprocesoarele moderne au lungimea cuvîntului de 32, 64 şi mai mulţi biţi.

Frecvenţa ceasului de sistem reprezintă numărul de impulsuri pe secundă pe care le produce generatorul de impulsuri de tact din componenţa dispozitivului de comandă. Frecvenţa ceasului de sistem se măsoară în **Megahertz** (1 *MHz* = = 10^6 *Hz*). Întrucît impulsurile de tact sincronizează executarea microoperaţiilor în toate dispozitivele microprocesorului, frecvenţa ceasului de sistem caracterizează rapiditatea microprocesorului.

Capacitatea magistralei reprezintă numărul de biţi ai succesiunilor binare transmise prin magistrală. De obicei, capacitatea magistralei de date este egală sau mai mare decît lungimea cuvîntului microprocesorului. În caz contrar, pentru a transmite un cuvînt, sînt necesare mai multe cicluri ale magistralei de date.

Capacitatea magistralei de adrese determină spaţiul de adrese care pot fi accesate direct de microprocesor. Astfel, un microprocesor avînd capacitatea magistralei de adrese de 16 biţi poate accesa 2^{16} locaţii ale memoriei interne, iar un microprocesor cu capacitatea magistralei de adrese de 32 de biţi poate accesa direct 2^{32} locaţii.

În ansamblu, lungimea cuvîntului, frecvenţa ceasului de sistem şi capacitatea magistralelor determină **capacitatea de prelucrare a microprocesorului**, care se măsoară în ***Mips*** – Megainstrucţiuni pe secundă. Pentru exemplificare, în *tabelul 6.2* sînt prezentate caracteristicile principale ale microprocesoarelor din familia *Intel*.

Tipul microprocesorului şi frecvenţa ceasului de sistem al calculatorului personal cu care lucraţi dvs. poate fi aflat cu ajutorul meniului contextual al pictogramei „My computer”. Amintim că acest meniu se afişează pe ecran cu ajutorul unui clic-dreapta pe pictograma respectivă.

Tabelul 6.2

Caracteristicile principale ale microprocesoarelor din familia *Intel*

Micro-procesor	Lungime cuvînt, *bit*	Frecvenţă, *MHz*	Capacitate de prelucrare, *Mips*
Pentium I	32	200	200
Pentium II	32	300	400
Pentium III	32	1400	1200
Pentium 4	32	3800	2500
Pentium D	64	3400	4000

Întrebări şi exerciţii

❶ Explicaţi destinaţia echipamentelor din componenţa unui microprocesor (*fig. 6.15*).

❷ Care sînt parametrii principali ai unui microprocesor? Explicaţi semnificaţia fiecărui parametru.

❸ Determinaţi tipul şi parametrii principali ai microprocesorului din componenţa calculatorului personal cu care lucraţi dvs.

Test de autoevaluare nr. 6

1. Indicați corespondența dintre denumirile unităților funcționale ale calculatorului (coloana din stînga) şi destinația acestora (coloana din dreapta):

(1) dispozitivul de intrare;	(a) efectuarea operațiilor aritmetice şi logice elementare;
(2) memoria;	(b) colorarea imaginilor şi memorarea lor pe discuri optice;
(3) dispozitivul aritmetic şi logic;	(c) extragerea datelor din calculator;
(4) dispozitivul de ieşire;	(d) furnizarea semnalelor de comandă necesare executării secvențiale a instrucțiunilor;
(5) dispozitivul central de comandă;	(e) introducerea documentelor în calculator şi corectarea greşelilor gramaticale;
(6) procesorul.	(f) înmagazinarea datelor inițiale, intermediare şi finale ale problemei, precum şi a instrucțiunilor care indică secvența calculelor;
	(g) prelucrarea automată a informației conform programului înmagazinat în memorie;
	(h) efectuarea calculelor aritmetice şi afişarea rezultatelor la ecran;
	(i) introducerea datelor din mediul exterior în calculator.

2. Numiți cel puțin cinci echipamente periferice pe care le cunoaşteți dvs.

3. Care dintre afirmațiile ce urmează sînt adevărate:

a) un calculator poate fi conceput fără ca să aibă o unitate de memorie externă;

b) memoria internă este un dispozitiv periferic;

c) memoria externă este mai lentă decît memoria internă;

d) unitatea de disc magnetic este o memorie internă;

e) un calculator poate fi conceput fără ca să aibă o unitate de memorie internă;

f) memoria externă este un dispozitiv periferic?

4*. Numiți destinația magistralei din componența calculatorului personal.

5*. Desenați schema-bloc a calculatorului personal. Care componente sînt obligatorii şi care opționale pentru funcționarea calculatorului?

6*. Este cunoscut faptul că instrucțiunile cu trei adrese conțin următoarele cîmpuri: *Codul instrucțiunii*, *Adresă operand 1*, *Adresă operand 2* şi *Adresă rezultat*. Explicați destinația fiecărui cîmp.

---

* Numai pentru profilul real.

7*. Explicați cum vor fi executate următoarele instrucțiuni cu trei adrese:

a) 01 101 153 342;

b) 04 508 391 216;

c) 03 751 852 031;

d) 02 450 709 011.

Operațiile aritmetice şi logice sînt codificate în felul următor: 01 – adunarea; 02 – scăderea; 03 – înmulțirea; 04 – împărțirea.

8*. Indicați tipul instrucțiunilor ce urmează (operaționale, de transfer, de salt sau de intrare-ieşire):

a) adunarea a două numere;

b) scrierea unui şir de octeți pe discul magnetic;

c) deplasarea unui cuvînt binar de la dreapta spre stînga;

d) citirea unui şir de octeți de pe discul optic;

e) compararea a două numere şi transferul controlului în funcție de rezultatul comparării;

f) înmulțirea a două numere;

g) încărcarea în registrul procesorului a unui număr din memoria internă.

9*. Se consideră că denumirea simbolică *X* semnifică locația 205, denumirea *Y* – locația 421, iar denumirea *S* – locația 783. Exprimați în limbajul de asamblare (vezi *tabelul 6.1*) următorul program:

```
01 205
02 783
04 421
03 205.
```

10*. Se consideră că denumirile simbolice *X*, *Y* şi *S* specifică, respectiv, locațiile 971, 583 şi 461. Translatați următoarele programe scrise în limbajul de asamblare (vezi *tabelul 6.1*):

```
INC Y
MEM S
INC X
MEM Y
INC S
```

11. Indicați tipul resurselor ce urmează (tehnice sau programate):

a) procesorul;

b) editorul de texte;

c) aplicația de calcul tabelar;

d) imprimanta;

e) aplicația de poştă electronică;

f) tastatura;

g) sistemul de operare;

h) vizualizatorul.

12. Care este diferența dintre memoriile externe cu acces direct şi acces secvențial?

13. Care dintre afirmațiile ce urmează sînt adevărate:

a) unitatea de memorie de bandă magnetică asigură accesul direct la înregistrări;

b) timpul de acces al unității de bandă magnetică este mai mare ca cel al unității de discuri magnetice;

c) unitatea de memorie de discuri magnetice asigură accesul secvențial la înregistrări;

d) pe o bandă magnetică înregistrările sînt grupate pe cilindri;

e) timpul de acces al unității de disc rigid este mai mic ca cel al unității de disc flexibil?

14. Care dintre următoarele afirmații sînt adevărate:

a) toate discurile optice au dezavantajul că odată ce au fost scrise nu mai pot fi rescrise;

b) în general, discurile magnetice rigide au capacități de stocare mai mari decît ale discurilor optice;

c) capacitatea de stocare a unui disc optic depinde de viteza cu care el se roteşte;

d) un disc magnetic flexibil are o capacitate de stocare mai mare decît un disc optic;

e) numărul ciclurilor de scriere/ştergere a discurilor optice reinscriptibile nu poate depăşi o anumită limită?

15*. Viteza liniară a discului optic este de 5,6 m/s, iar densitatea de înregistrare a informației este de 512 *Kocteți*/m. Determinați viteza de transmisie a datelor de la unitatea de disc optic la unitatea centrală.

16. Alegeți din lista ce urmează parametrii ce caracterizează un vizualizator:

a) rezoluția;

b) numărul de pagini ce pot fi afişate într-o secundă;

c) diagonala ecranului;

d) numărul de culori pe care le poate afişa;

e) viteza de imprimare.

17. Alegeți din lista ce urmează parametrii ce caracterizează o imprimantă:

a) rezoluția;

b) numărul de culori pe care le poate imprima;

c) diagonala imprimantei;

d) numărul de instrucțiuni pe secundă;

e) viteza de imprimare.

18. Indicați parametrii de bază ce caracterizează un calculator.

19*. Indicați parametrii de bază ai microprocesorului.

Capitolul 7

REȚELE DE CALCULATOARE

7.1. Introducere în rețele

Odată cu extinderea domeniilor de aplicare a calculatoarelor, a crescut şi numărul utilizatorilor ce doreau să aibă acces la mijloace eficiente de prelucrare şi stocare a unor informaţii comune.

De exemplu, în cazul proiectării unei clădiri, mai mulţi specialişti – arhitectul, inginerul, pompierul etc. doresc să aibă acces şi să introducă, dacă e necesar, modificări în desenele tehnice ale construcţiei în curs de elaborare. În cazul unei companii de transporturi aeriene biletele la una şi aceeaşi cursă pot fi vîndute de mai multe agenţii aflate în oraşe 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*).

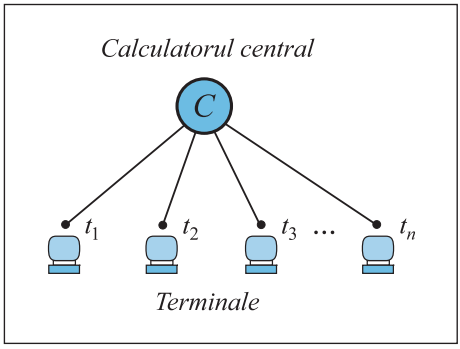


Fig. 7.1. Sistem centralizat de calcul

De regulă, un terminal constă dintr-un vizualizator, o tastatură şi, dacă e necesar, o imprimantă. Neajunsul principal al unui sistem centralizat de prelucrare a datelor este fiabilitatea redusă şi utilizarea ineficientă a resurselor de calcul.

Cu timpul, a apărut tendința de trecere de la sistemele centralizate la instalarea de calculatoare la fiecare utilizator şi asigurarea unor legături de comunicaţie eficientă între ele (*fig. 7.2*).

Numim rețea de calculatoare o mulțime de calculatoare ce pot schimba informaţii prin intermediul unei structuri de comunicaţie.

Calculatoarele unei rețele se conectează la structura de comunicaţie prin intermediul unor unități de intrare-ieşire dedicate, numite **adaptoare de rețea**.

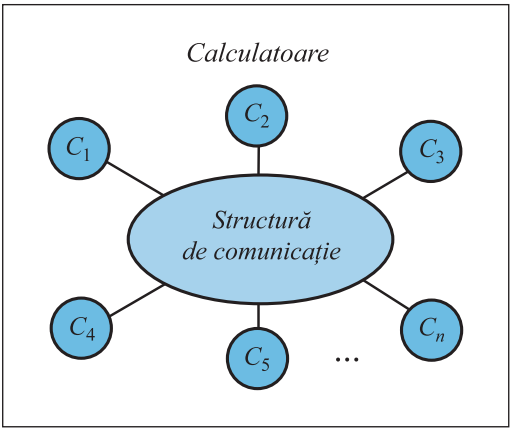


Fig. 7.2. Rețea de calculatoare

Evident, în cadrul unei rețele fiecare calculator, mai exact, fiecare adaptor de rețea, are o adresă unică, denumită **adresă de rețea**.

De exemplu, o rețea de calculatoare poate fi construită utilizînd 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 şi un demodulator pentru operaţia inversă. Dispozitivul respectiv poartă denumirea de **modem** (**mo**dulator-**dem**odulator). 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 sînt asemănătoare celor telefonice şi asigură o capacitate de transmisie de pînă la 1 *Mbit*/s. **Cablurile coaxiale**, asemănătoare celor din rețelele de televiziune prin cablu, asigură o capacitate de transmisie de pînă la 1 *Gbit*/s. **Cablul optic** constă din fibre de sticlă sau din plastic transparent, acoperite cu un înveliş de protecţie. Semnalul optic, emis de o sursă laser, se propagă prin fibră şi este recepționat de o celulă fotosensibilă. Capacitatea de transmisie a unui cablu optic poate ajunge la valoarea de 1 *Tbit*/s.

Liniile cu microunde sînt formate din staţii de retransmisie ce operează în banda de unde centimetrice. Pe Pămînt aceste staţii se amplasează în raza vizibilităţii directe a antenelor, la o distanţă de 40–50 de kilometri una de alta. În cazul liniilor cosmice staţiile respective se amplasează pe sateliţi. Capacitatea de transmisie a liniilor cu microunde este de ordinul 10 *Gbiţi*/s.

În funcţie de **aria de răspîndire** 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 răspîndire (pînă la 2 km) şi deservesc o singură instituție. Rețelele locale sînt formate, de regulă, din calculatoarele care se află în aceeaşi clădire sau într-un grup de clădiri. De obicei, ca linii de transmisie se utilizează cablurile cu fire torsadate şi cablurile coaxiale.

Rețelele regionale acoperă aria unui oraş sau a unui sector. Liniile de comunicație se realizează prin cabluri coaxiale sau staţii 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 şi liniile cu microunde (terestre sau prin satelit).

Avantajul principal al rețelelor constă în **partajarea** sau, cu alte cuvine, **utilizarea în comun** a datelor, a programelor şi a calculatoarelor din rețea.

De exemplu, în cazul unei rețele locale pot fi partajate fişierele, discurile de capacitate mare, imprimantele, cititoarele de desene şi alte periferice. Evident, fiind accesibile pentru mai mulți utilizatori, echipamentele periferice respective vor fi utilizate mai eficient. Totodată, specialiştii instituției în cauză pot lucra în echipă asupra unor proiecte comune: bugetul anual, planul de vînzări, 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 ştiinţific foarte costisitor. Pe baza rețelelor examinate sînt create diverse servicii: transferul de fişiere, poşta electronică, difuzarea noutăților, conversații pe grupuri de interese, jocuri electronice, publicitate, transferul banilor etc.

Întrebări şi exerciții

❶ Numiți factorii care au contribuit la apariția rețelelor de calculatoare.

❷ Care sînt neajunsurile sistemelor centralizate de calcul?

❸ Numiți componentele principale ale unei rețele de calculatoare.

❹ Explicați destinația structurii de comunicație.

❺ Care sînt 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 (≈800 *Gbiți*) prin liniile de comunicație pe care le cunoaşteți dvs.

❿ Determinați tipul liniilor de comunicație din structura rețelelor cu care lucrați dvs.

⓫ Cum se clasifică rețelele în funcție de aria de răspîndire?

⓬ Determinați tipul rețelei (locală, regională sau globală) cu care lucrați dvs.

⓭ Care sînt avantajele rețelelor de calculatoare? Ce servicii oferă o rețea de calculatoare?

7.2. Tehnologii de cooperare în rețea

Resursele unei rețele de calculatoare sînt 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 şi programelor ce rulează pe ele.

Numim tehnologie de cooperare modul cum este organizată funcționarea în comun a calculatoarelor şi programelor din rețea.

Cel mai frecvent în rețelele de calculatoare se utilizează tehnologiile client-server şi 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 doreşte să aibă acces la aceste resurse se numeşte **client**. Pentru a utiliza resursele respective, orice client trimite serverului o cerere. Serverul analizează cererile primite şi, în funcție de statutul fiecărui client, le acceptă sau le respinge (*fig. 7.3*).

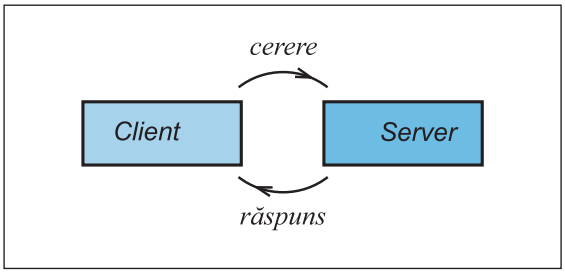


Fig. 7.3. Modelul client-server

Evident, calculatorul care gestionează fişiere comune se va numi **server de fişiere**, iar cel care gestionează imprimantele – **server de imprimare**.

Calculatorul care gestionează liniile de comunicație, alte resurse comune şi, posibil, asigură accesul la rețele externe, se numeşte **server de rețea**. De regulă, anume pe acest calculator rulează şi sistemul de operare al rețelei. Celelalte calculatoare din rețea au caracteristici mai modeste şi se numesc **staţii de lucru**.

Avantajele principale ale tehnologiei client-server sînt:

– 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 sînt complicate şi necesită calculatoare performante.

În **tehnologia egal-la-egal** funcțiile tuturor calculatoarelor din rețea sînt identice. Fiecare calculator din rețea funcționează atît ca server, cît şi ca stație de lucru, oferind pentru uzul public resursele de care dispune, în mod curent, unele fişiere de pe discul fix, unitatea de disc optic, imprimanta etc. Fiind simplă, tehnologia dată se foloseşte 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 şi pentru organizarea lucrului în comun a două sau a mai multe programe.

Programul în curs de execuţie care oferă servicii se numeşte 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 funcţii:

– asigură protecţia şi securitatea datelor;

– recepţionează şi, dacă clientul are autorizaţiile respective, execută cererile de modificare a datelor;

– recepţionează cererile de citire a datelor şi, în funcţie de statutul clientului, permite sau interzice accesul la datele respective;

– completează un registru în care înscrie toate operaţiile efectuate în baza de date.

Programul client asigură interacţiunea utilizatorului cu baza de date şi realizează următoarele funcţii:

– oferă utilizatorului o interfaţă simplă şi comodă;

– verifică şi editează datele introduse de utilizator;

– adresează cereri programului server;

– afişează informaţiile primite din baza de date.

Programele server şi 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 şi programul server se realizează prin structura de comunicaţie (*fig. 7.2*). Calculatorul pe care rulează un program server se numeşte **gazdă** (în engleză *host*).

De obicei, în reţelele locale programul client se execută pe staţiile de lucru, iar programele server rulează pe serverul de reţea. În cazul reţelelor regionale sau globale pe fiecare calculator performant rulează mai multe programe server ce oferă diverse servicii programelor client de pe alte calculatoare.

Întrebări şi exerciţii

❶ Explicaţi termenul *tehnologii de cooperare în reţea*. Ce tehnologii de cooperare în reţea cunoaşteţi?

❷ Cum este organizată funcţionarea calculatoarelor din reţea în cazul tehnologiei client-server?

❸ Care sînt avantajele şi dezavantajele tehnologiei client-server?

❹ Explicaţi destinaţia serverelor de reţea, a serverului de fişiere, a serverului de imprimare şi a staţiei de lucru.

❺ Cum este organizată funcţionarea calculatoarelor din reţea în cazul tehnologiei egal-la-egal? Care sînt avantajele şi dezavantajele acestei tehnologii?

❻ Determinaţi tehnologiile de cooperare realizate în reţeaua cu care lucraţi dvs. Există oare în reţea un server de fişiere şi/sau un server de imprimare?

❼ Există în reţeaua cu care lucraţi dvs. un server de reţea? Argumentaţi răspunsul.

❽ Este oare realizată în reţeaua cu care lucraţi dvs. tehnologia egal-la-egal? Argumentaţi răspunsul.

❾ Cum interacționează programele în curs de execuție în cazul tehnologiei client-server?

❿ Numiți funcțiile unui program server şi ale unui program client.

⓫ Explicați termenul *calculator-gazdă*.

⓬ O companie de transporturi aeriene a deschis agenții de vînzare a biletelor în mai multe orașe din țară. Datele despre toate rutele aeriene şi locurile disponibile se păstrează în calculatorul din sediul central al companiei.
Care tehnologii de rețea ar asigura lucrul eficient al agențiilor de vînzare a biletelor? Sînt oare necesare programe server şi programe client? Ce funcții ar realiza aceste programe?

⓭ 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?

⓮ 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 şi un radiomodem);
g) laboratorul de informatică.

7.3. Topologia şi arhitectura rețelelor

Structura de comunicație a unei rețele (*fig. 7.2*) asigură transferul de date între calculatoare. De obicei, datele de transmis sînt 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 obişnuit care circulă într-un sistem tradițional de poştă. Drumul parcurs de un pachet de date depinde de topologia rețelei.

Numim topologie a rețelei configurația geometrică a legăturilor între calculatoare.

Topologiile concrete ale rețelelor actuale sînt formate prin utilizarea structurilor de bază: stea, inel, magistrală, distribuită etc. (*fig. 7.4*).

În cazul **topologiei stea** legătura între 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, efectuînd dispecerizarea pachetelor de date. Evident, defectarea calculatorului prin-

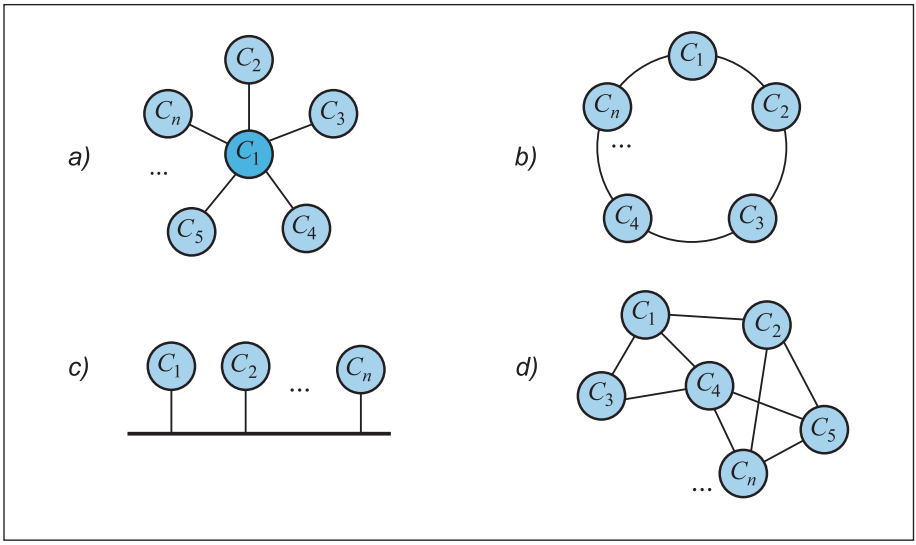


Fig. 7.4. Topologii de rețea:
a – stea; *b* – inel; *c* – magistrală; *d* – distribuită

cipal întrerupe funcționarea întregii rețele. Prin urmare, calculatorul principal trebuie să fie foarte fiabil.

În **topologia inel** legăturile între calculatoare formează o buclă închisă. Pachetul expediat de calculatorul C_i este transmis calculatorului C_{i+1}, care la rîndul său îl transmite calculatorului C_{i+2} etc. pînă cînd pachetul ajunge la calculatorul destinatar C_j. Întrucît 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 sînt conectate toate calculatoarele. Fiecare calculator „spionează" magistrala şi interceptează pachetele adresate lui. Orice calculator C_i poate expedia un pachet numai atunci cînd magistrala este liberă.

Rețelele bazate pe topologia de tip magistrală sînt foarte fiabile, întrucît comunicarea între calculatoarele C_i, C_j va avea loc chiar şi 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 rămîne funcțională chiar dacă unul sau mai multe calculatoare şi linii de comunicație nu funcționează.

De regulă, topologiile de tip stea, inel sau magistrală se utilizează în cazul rețelelor locale. Rețelele regionale şi cele globale au o topologie distribuită. Integrarea rețelelor locale în rețele regionale şi rețele globale se face conform topologiilor de bază din *figura 7.4,* fiecare nod C_1, C_2, ..., C_n reprezentînd 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 şi simplu, protocol.

Un **protocol** defineşte modul de adresare a calculatoarelor, lungimea şi componenţa pachetelor de date, algoritmul de depistare şi corectare a erorilor, modul de conectare fizică a adaptoarelor şi cablurilor de reţea etc. La apariţia primelor reţele de calculatoare fiecare producător de echipamente de calcul avea propriile sale protocoale de comunicaţie, ceea ce făcea imposibilă interconectarea calculatoarelor de provenienţe 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 internaţionale sînt elaborate de către Organizaţia Internaţională pentru Standardizare (*ISO – International Standards Organisation*). Un alt organism internaţional care joacă un rol important în standardizarea în domeniile electronicii şi tehnicii de calcul este Institutul Inginerilor Electricieni şi Electronişti (*IEEE – Institute of Electrical and Electronics Engineers*).

Numim arhitectură a reţelei modul în care este concepută reţeaua: topologia, protocoalele de comunicaţie, tehnologiile de cooperare în reţea.

În continuare prezentăm cele mai răspîndite arhitecturi de reţele.

Ethernet (reţea în eter) – reţele 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 *Mbiţi*/s. Arhitectura a fost elaborată de firmele *XEROX*, *Intel* şi *DEC*.

Token–Ring (inel cu jeton) – reţele 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 *Mbiţi*/s. Arhitectura a fost elaborată de firma *IBM*.

DATAKIT – reţele locale, regionale sau globale elaborate de firma *Bell Laboratories*. Sub aspect topologic constă dintr-o mulţime de stele interconectate. În cazul cablului cu fibre optice se atinge viteza de transmisie de 1,5 *Gbiţi*/s.

SNA (*Sistem Network Arhitecture*) – o arhitectură elaborată de firma *IBM* pentru reţelele locale, regionale şi globale. Protocoalele acestei arhitecturi au stat la baza standardelor *ISO*. Topologia iniţială era de tip stea, în prezent este o topologie distribuită, suportînd şi reţele locale.

ARPANET – o arhitectură concepută de mai multe universităţi şi corporaţii sub egida Ministerului Apărării al SUA (*Advanced Research Projects Agency*). Arhitectura dată se bazează pe o topologie distribuită şi foloseşte diferite linii de comunicaţie, de la liniile telefonice pînă la liniile cu microunde prin satelit. Liniile respective conectează supercalculatoare separate şi diverse reţele locale sau regionale, răspîndite pe aproape jumătate din suprafaţa terestră. Arhitectura *ARPANET* include următoarele protocoale:

– protocolul ***IP*** (*Internet Protocol*) destinat interconectării reţelelor locale, regionale şi globale;

– protocolul serviciilor bazate pe conexiuni, numit ***TCP*** (*Transmission Control Protocol*);

– un protocol de transfer de fişiere, numit ***FTP*** (*File Transfer Protocol*);

– un protocol pentru poşta electronică, numit ***SMTP*** (*Simple Mail Transfer Protocol*);

– un protocol de ataşare de la distanţă a calculatoarelor, numit ***TELNET***.

Pe baza arhitecturii *ARPANET* a fost concepută reţeaua globală de calculatoare *Internet*.

Întrebări şi exerciţii

❶ Explicaţi termenul *pachet de date*. Ce informaţii conţine un pachet de date?

❷ Explicaţi termenul *topologia reţelei*.

❸ Care sînt topologiile de bază ale reţelelor? Numiţi avantajele şi dezavantajele fiecărei topologii de bază.

❹ Explicaţi cum se transmit pachetele de date în următoarele topologii de bază: stea, inel, magistrală, distribuită.

❺ Precizaţi drumul pe care îl va parcurge un pachet trimis de calculatorul C_2 calculatorului C_4 (*fig. 7.4a*).

❻ Găsiţi drumul pe care îl parcurg pachetele expediate de calculatorul C_2 calculatorului C_4 (*fig. 7.4b*).

❼ Cîte 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 întîmpla cu reţelele din *figura 7.4* dacă iese din funcţie calculatorul C_1? Dar dacă iese din funcţie calculatorul C_2?

❾ Determinaţi topologia reţelei locale cu care lucraţi dvs. Enumeraţi avantajele şi dezavantajele acestei topologii.

❿ Care este destinaţia unui protocol? Ce norme conţine un protocol?

⓫ Argumentaţi necesitatea standardizării protocoalelor. Cine elaborează standardele respective?

⓬ Aflaţi protocoalele utilizate de reţeaua locală cu care lucraţi dvs.

⓭ Explicaţi modul cum se integrează reţelele locale în reţele regionale şi reţele globale. Desenaţi topologia reţelelor regionale sau globale la care aveţi acces dvs.

⓮ Explicaţi termenul *arhitectura reţelei*.

⓯ Daţi exemple de arhitecturi ale reţelelor locale, regionale şi globale.

⓰ Caracterizaţi arhitectura reţelei cu care lucraţi dvs.

⓱ Enumeraţi protocoalele utilizate în arhitectura *ARPANET*. Sînt oare aplicate aceste protocoale în reţeaua cu care lucraţi dvs.?

7.4. Reţeaua *Internet*

Reţeaua globală *Internet* se bazează pe o topologie distribuită şi include calculatoare separate, subreţele locale, regionale sau globale (*fig. 7.5*).

Interconectarea reţelelor se realizează cu ajutorul unor echipamente dedicate de reţea, denumite porţi sau rutere. **Poarta** (*gateway*) este un calculator specializat destinat interconectării a două reţele ce utilizează protocoale total diferite de co-

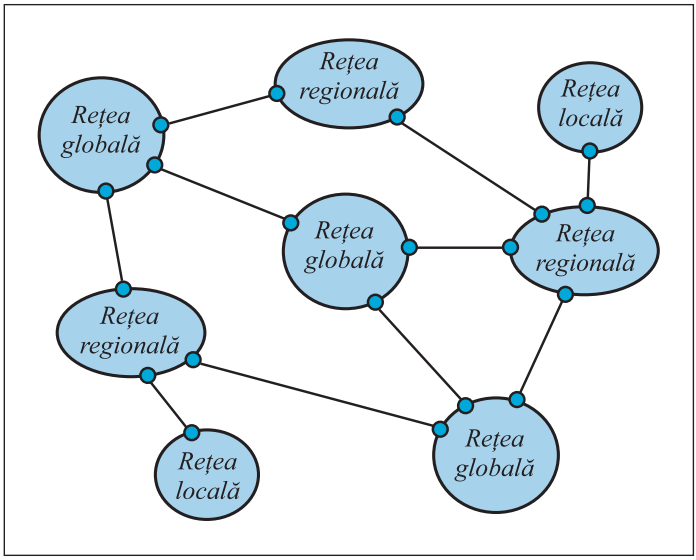


Fig. 7.5. Topologia *Internetului*

municaţie. **Ruterul** (*router*) este un calculator dedicat care interconectează rețelele ce folosesc protocoale identice şi sînt 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 şi adrese simbolice.

O **adresă numerică** este formată din 32 de cifre binare (4 octeți) şi are structura prezentată în *figura 7.6*.

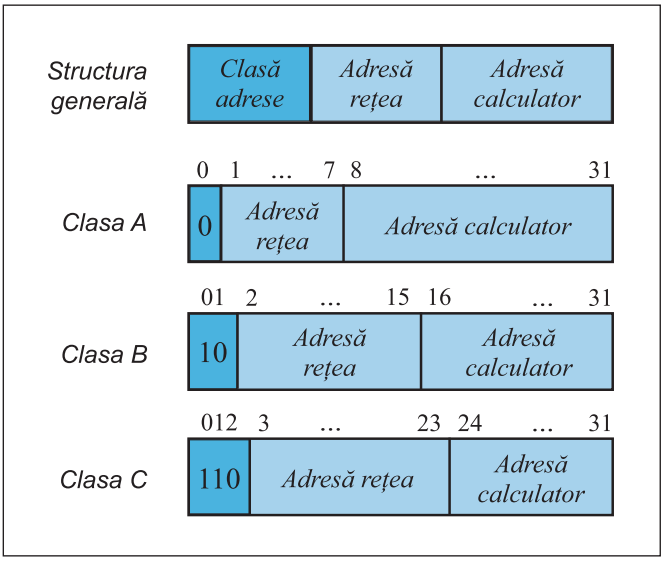


Fig. 7.6. Structura adreselor numerice

Întrucît *Internetul* este o „rețea de rețele”, adresa numerică conține adresa subrețelei (cîmpul *Adresă Rețea*) şi adresa calculatorului în cadrul subrețelei (cîmpul *Adresă Calculator*). În funcție de numărul maximal de calculatoare pe care le poate identifica în cadrul unei subrețele, adresele se împart în clasele *A*, *B* şi *C*. Adresele din clasele *D* şi *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* sînt distribuite rețelelor mari, în special rețelelor globale. O astfel de rețea poate include circa 16 milioane de calculatoare. Adresele din clasa *B* sînt distribuite rețelelor medii, în special rețelelor regionale. O astfel de rețea poate include circa 65 mii de calculatoare. Adresele din clasa *C* sînt rezervate rețelelor relativ mici, care includ pînă la 256 de calculatoare. Fiecare adresă *Internet* este unică. Adresele sînt atribuite calculatoarelor de **Centrul Informațional al Rețelei** (*Network Information Center*).

De exemplu, numărul binar

10010010 00110011 00001001 11110111

este o adresă de clasa *B* şi identifică calculatorul numărul

00001001 11110111

din cadrul rețelei

010010 00110011.

Pentru comoditate, adresele binare se exprimă în formă zecimală, transformînd 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 forma zecimală adresa respectivă va fi 146.51.9.247.

Adresele în formă zecimală şi cu atît mai mult cele în formă binară sînt incomode pentru publicul larg. Din acest motiv, mai frecvent sînt utilizate adresele simbolice.

O **adresă simbolică** este formată din numele calculatorului-gazdă şi nume de domenii separate prin puncte. **Domeniul** reprezintă un grup de calculatoare organizate tematic sau geografic. Orice domeniu poate fi împărţit în subdomenii, ajungîndu-se astfel la o structură ierarhică. Numele de domenii se indică în ordinea creşterii ariei de cuprindere.

De exemplu, adresele simbolice

```
c1.lme.ch.md
```

```
c5.lme.ch.md
```

specifică calculatoarele c1 şi c5 din domeniul lme (Liceul „Mihai Eminescu”).

Adresele simbolice

```
c1.lic.ch.md
```

```
c9.lic.ch.md
```

specifică calculatoarele c1 şi c9 din domeniul lic (Liceul „Ion Creangă”). Domeniile lme şi lic sînt subdomenii ale domeniului ch (Chişinău). La rîndul 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 şi decan din domeniul ase (Academia de Studii Economice). Domeniul ase este un subdomeniu al domeniului men (Ministerul Educaţiei Naţionale), iar men este un subdomeniu al domeniului ro (România).

În mod normal, domeniul de cel mai înalt nivel este ţara (md, ro, us etc.) sau tipul instituţiei (com – comercială, mil – militară, edu – de educaţie etc.).

Relaţiile de incluziune între domenii pot fi reprezentate cu ajutorul diagramelor *Euler*, frecvent utilizate în teoria mulţimilor. Pentru exemplificare, în *figura 7.7* sînt prezentate astfel de diagrame pentru unele adrese simbolice din domeniul md.

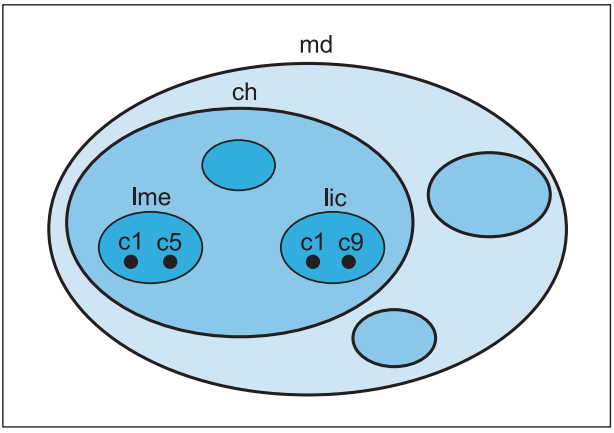


Fig. 7.7. Diagrame *Euler* pentru adrese simbolice

Pentru a **transforma** adresele simbolice în adrese numerice şi invers, în fiecare domeniu există un **server de nume** (*name server*). Acest program gestionează domeniul respectiv fără intervenţia serverelor ierarhic superioare. Prin urmare, în *Internet* nu există un calculator central care ar fi unicul responsabil de cele aproximativ 10^{10} adrese de reţea.

În cazul exemplului de mai sus (*fig. 7.7*), adresele calculatoarelor c1 şi c5 sînt procesate în serverul lme, iar adresele calculatoarelor c1 şi c9 – de serverul lic. Serverul ch va lucra numai cu adresele serverelor lic şi lme, fără să se implice în procesarea adreselor de calculatoare din aceste domenii. În mod similar, serverul men nu procesează adresele calculatoarelor rector şi decan, lăsînd acest lucru în obligaţia serverului ase.

Întrebări şi exerciţii

❶ Desenaţi topologia reţelei *Internet*. Cum se interconectează subreţelele în cadrul *Internetului*?

❷ Care este destinaţia unei porţi? Dar a unui ruter?

❸ Cum se identifică calculatoarele în *Internet*? Care sînt avantajele şi dezavantajele adreselor numerice? Dar ale adreselor simbolice?

❹ Care sînt clasele de adrese numerice? Pentru ce este necesară o astfel de clasificare?

❺ Explicaţi destinaţia cîmpurilor *Adresă reţea* şi *Adresă calculator* ale unei adrese numerice.

❻ Cine atribuie adrese calculatoarelor din *Internet*?

❼ Determinaţi clasele următoarelor adrese. Precizaţi adresa subreţelei şi adresa calculatorului în subreţea.

a) 45.201.19.63;
b) 201.165.213.91;
c) 154.36.79.200;
d) 192.109.58.170;
e) 15.21.207.250;
f) 217.15.69.113.

❽ Care sînt criteriile de grupare a calculatoarelor în domenii? Numiţi domeniile de cel mai înalt nivel.

❾ Sînt date următoarele adrese simbolice:

a) c1.lme.ch.md;
b) c3.lme.ch.md;
c) c1.lic.ch.md;
d) director.lic.ch.md;
e) elev1.lic.ch.md;
f) c4.lme.ch.md;
g) c5.lme.ch.md;
h) c9.lic.ch.md;
i) prof.lic.ch.md;
j) elev4.lic.ch.md.

Precizaţi domeniile de calculatoare şi relaţiile de incluziune între domenii. Desenaţi 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;
b) decan.ase.men.ro;
c) student.info.ase.men.ro;
d) rector.ase.met.md;
e) decan.ase.met.md;
f) student.cib.met.md.

⓫ Care este destinația unui server de nume? Ce adrese sînt 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 Aflați adresa numerică și adresa simbolică a 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 accesul 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 încît originalul și copia sînt 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 poştă electronică (*electronic mail* sau, prescurtat, *e-mail*) a copiat modul de funcţionare a poştei obişnuite.

Scrisoarea electronică, denumită **mesaj** (*message*) include:

– adresa destinatarului;
– subiectul, exprimat în cîteva cuvinte;
– adresa expeditorului;
– textul scrisorii;
– fişiere ataşate opţional.

Fişierele ataşate pot fi de orice natură: texte, imagini, programe etc.

Scrisorile sînt depuse în fişiere speciale, denumite **cutii poştale** (*mail box*). Adresa unei cutii poştale are forma:

```
<nume cutie>@<Adresă calculator>,
```

unde:

<nume cutie> este denumirea cutiei poştale, 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 poştală.

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 sînt transmise prin reţea de serverele de poştă care au rolul oficiilor şi centrelor poştale tradiţionale.

Serviciul de poştă electronică este foarte popular datorită avantajelor sale incontestabile, şi anume: viteză, posibilitatea de a ataşa la scrisori fişiere de orice natură, facilităţi avansate de redactare.

Cel mai modern serviciu de prezentare şi căutare a informaţiilor în *Internet* este **serviciul WWW** (*World Wide Web* – Pînza Mondială de Păianjen). În acest serviciu informaţia este prezentată în formă de pagini Web.

Pagina Web este un fişier scris în limbajul HTML (*Hypertext Markup Language* – Limbaj pentru marcarea hipertextului) şi poate conţine, în afară de informaţii propriu-zise, referinţe la alte pagini Web. Paginile referite se pot afla pe acelaşi calculator sau pe calculatoare situate în diverse puncte geografice (*fig. 7.8*).

Tehnologia de cooperare în cadrul serviciului WWW este de tipul client-server. Utilizatorul care doreşte să ofere publicului larg anumite informaţii instalează pe

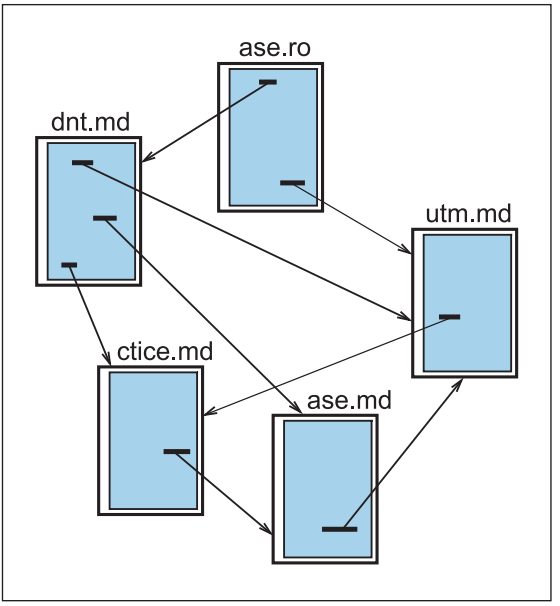


Fig. 7.8. Pagini Web

calculatorul său un program server şi elaborează una sau mai multe pagini Web. Serverul interceptează cererile sosite de la alte calculatoare şi asigură accesul la paginile respective. Calculatorul pe care sînt instalate paginile Web şi serverul WWW se numeşte **site** (*site* – sediu, reşedinţă).

Programul client asigură transferul şi afişarea pe ecran a paginilor Web citite de pe diverse calculatoare din *Internet*. Imediat după pornire, el afişează propria pagină de referinţă (*home page* – pagina de acasă) şi aşteaptă indicaţiile utilizatorului. Cînd utilizatorul activează o referinţă, programul client stabileşte o conexiune cu serverul Web şi copiază de la el pagina specificată în referinţă. Pagina copiată este afişată pe ecran.

În continuare, utilizatorul activează o altă referinţă, programul client stabileşte din nou conexiunea cu un calculator din reţea, din nou se va citi o pagină Web etc. Cu alte cuvinte, utilizatorul „răsfoieşte" paginile Web de pe diverse calculatoare, indiferent de poziţia lor geografică. Din acest motiv programele client sînt numite **programe de răsfoire** sau **programe de explorare** (în engleză *browser* sau *explorer*).

În cadrul serviciului WWW resursele reţelei 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>/<fişier>
```

unde:

<protocol> – specifică denumirea protocolului pentru transferul datelor prin reţea;

<Adresă simbolică> – adresa calculatorului ce conţine fişierul respectiv;

<port> – portul de acces (opţional);

<cale>/<fişier> – specificarea fişierului.

Pentru exemplificare, prezentăm cîteva adrese URL ce conţin informaţii interesante:

http://www.edu.md – site-ul Ministerului Educaţiei şi Tineretului, Moldova;

http://www.ctice.edu.md – site-ul Centrului de Tehnologii Informaţionale şi Comunicaţionale în Educaţie, Moldova;

http://www.dnt.md – site-ul Asociaţiei *Dynamic Network Tehnologies*, Moldova;

http://www.itc.ro/museum/museum.html – site-ul Muzeului Naţional de Artă, România;

http://www.nmsi.ac.uk – site-ul Muzeului de Ştiinţă şi Industrie, Marea Britanie;

http://www.nasa.gov – site-ul agenţiei *NASA*, Statele Unite ale Americii.

Amintim că notaţia http specifică protocolul de transfer al hypertextelor (*Hypertext Transfer Protocol*).

În prezent, numărul fişierelor din reţeaua *Internet* este de ordinul miliardelor. Evident, nici nu poate fi vorba de căutarea informaţiei necesare prin citirea individuală a fiecărui fişier. Pentru a simplifica căutarea informaţiei, în cadrul reţelei *Internet* au fost create servere de căutare (*search engine*).

Serverul de căutare este un calculator puternic care explorează încontinuu reţeaua şi citeşte paginile Web sau alte informaţii prezentate publicului larg. Acestea sînt clasificate în funcţie de datele pe care le conţin, iar adresele lor sînt reţinute în baza de date de pe server.

Programul client adresează serverului de căutare o cerere în care indică de ce fel de informaţii are nevoie. Serverul interoghează baza de date şi transmite clientului o listă de adrese la care pot fi găsite informaţiile cerute.

Pentru exemplificare amintim serverele de căutare frecvent utilizate:

http://www.yahoo.com – serverul YAHOO (*Yet Another Hierarhicaly Organizated Oracle* – încă un oracol organizat ierarhic) al companiei *Yahoo! Inc.*;

http://www.google.com – serverul GOOGLE al corporaţiei *Google Inc.*;

http://www.bing.com – serverul BING al corporaţiei *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 şi exerciţii

❶ Care este gama de servicii oferite de *Internet*? Cum colaborează calculatoarele din reţea în procesul prestării unui serviciu?

❷ Care este destinaţia serviciului *FTP*? Ce fişiere pot fi transferate prin acest serviciu?

❸ Studiaţi programul *FTP* instalat pe calculatorul dvs. Copiaţi cîteva fişiere de pe serverele *FTP anonymous* sau de pe alte servere la care aveţi acces.

❹ De ce depinde viteza de transfer al fişierelor? Determinaţi viteza de transfer a cîtorva fişiere de pe serverele *FTP* aflate în Republica Moldova, România şi Statele Unite ale Americii.

❺ Cum se specifică adresele în cadrul serviciilor de poştă electronică?

❻ Explicaţi cum interacţionează calculatoarele client şi serverele de poştă în cadrul serviciului respectiv.

❼ Ce informaţii conţine o scrisoare electronică? Care dintre ele sînt opţionale?

❽ Explicaţi cum se formează o adresă poştală. Aflaţi adresele poştale ale prietenilor dvs.

❾ Studiaţi programul de poştă electronică instalat pe calculatorul la care lucraţi dvs. Verificaţi dacă acest program oferă următoarele facilităţi:
– utilizarea mai multor cutii poştale;
– clasificarea şi păstrarea scrisorilor în dosare;
– elaborarea scrisorilor conform unor şabloane;
– cifrarea şi descifrarea corespondenţei;
– semnalarea momentelor cînd a sosit corespondenţa;
– verificarea faptului că scrisorile expediate au ajuns la destinaţie;
– utilizarea semnăturilor electronice.

❿ Care sînt avantajele poştei electronice? Poate înlocui acest serviciu poşta tradiţională?

⓫ Formaţi grupe de cîte patru utilizatori ai poştei electronice şi determinaţi experimental viteza cu care se transmite corespondenţa.

⓬ Ce informaţii conţine paginaWeb? Cum formează aceste pagini o „pînză de păianjen”?

⓭ Care este destinaţia unui server Web? Dar a unui client Web? Cum interacţionează aceste programe?

⓮ Explicaţi modul de funcţionare a unui client Web. Cum găseşte acest program paginile Web amplasate pe diferite calculatoare?

⓯ Explicaţi modul de specificare a resurselor în *Internet* cu ajutorul adreselor URL. Care este semnificaţia cîmpurilor acestor adrese?

⓰ Studiaţi programul de explorare a *Internetului* instalat pe calculatorul cu care lucraţi dvs. Ce facilităţi oferă acest program? Citiţi paginile Web adresele cărora sînt indicate în acest paragraf.

⓱ Care este destinaţia unui server de căutare? Ce servicii oferă un astfel de server?

⓲ Găsiţi cu ajutorul unui server de căutare furnizorii de servicii *Internet* din Republica Moldova.

⓳ În afară de serviciile studiate în acest paragraf, reţeaua *Internet* oferă şi alte servicii, cum ar fi: *Archie*, *Gopher*, *WAIS*, buletine de ştiri, discuţii etc. Folosind un server de căutare, aflaţi informaţia respectivă despre aceste servicii.

⓴ Elaboraţi paginile Web ale clasei şi ale liceului dvs.

Test de autoevaluare nr. 7

1. Estimați durata de transmisie a unui film video în formă arhivată (≈750 *Mbiți*) prin următoarele linii de comunicație:
a) linie telefonică de 36 *Kbiți/s*;
b) cablu cu fire torsadate de 1 *Mbit/s*;
c) cablu coaxial de 1 *Gbit/s*;
d) cablu optic de 1 *Tbit/s*.

2. Care dintre afirmațiile ce urmează sînt adevărate:
a) rețelele locale acoperă aria unei localități;
b) rețelele globale acoperă suprafața unei țări, suprafața unui continent sau chiar suprafața mai multor continente;
c) în general, rețelele regionale deservesc localitățile unei singure companii;
d) de obicei, rețelele locale deservesc o singură instituție?

3. Numiți componentele principale ale unei rețele de calculatoare.

4. Care dintre afirmațiile ce urmează sînt adevărate:
a) în tehnologia egal-la-egal funcțiile tuturor calculatoarelor din rețea sînt identice;
b) serverul unei rețele are aceleaşi funcții ca şi oricare alt calculator din rețea;
c) în tehnologia client-server o resursă comună este gestionată de un calculator dedicat;
d) orice calculator-client poate accesa datele de pe oricare alt calculator din rețea?

5* . Care dintre afirmațiile ce urmează sînt adevărate:
a) în topologia stea legătura între două calculatoare ale rețelei are loc prin intermediul calculatorului central;
b) în topologiile distribuite legătura între două calculatoare ale rețelei are loc prin intermediul magistralei;
c) în topologia magistrală legătura între două calculatoare ale rețelei are loc prin intermediul calculatorului central;
d) în topologia inel toate calculatoarele au funcții identice?

6*. Care este destinația unui protocol de rețea?

7*. Explicați semnificația termenului *arhitectura rețelei*.

8. Desenați topologia rețelei *Internet*.

9. Care dintre afirmațiile ce urmează este adevărată:
a) Internetul este o rețea locală ce are acces la alte rețele locale, regionale sau globale;
b) Internetul reprezintă totalitatea rețelelor dintr-o țară;
c) Internetul este o rețea a marilor companii internaționale;
d) Internetul reprezintă o rețea distribuită, formată din rețele globale, regionale şi locale interconectate între ele;
e) *Internetul* este format din toate calculatoarele lumii, interconectate între ele?

10. Cum se identifică calculatoarele în *Internet*?

11*. Determinați clasa adresei numerice *Internet* 214.121.216.109. Precizați adresa subrețelei şi adresa calculatorului în subrețea.

---

* Numai pentru profilul real.

12. Desenați o diagramă *Euler* pentru adresele simbolice ce urmează. Precizați domeniile de calculatoare şi relațiile respective de incluziune.

a) directie.orhei.md;
b) contabilitate.orhei.md;
c) presedinte.orhei.md;
d) directie.cahul.md;
e) contabilitate.cahul.md;
f) presedinte.calarasi.md.

13. Indicați corespondența între denumirile serviciilor *Internet* (coloana din stînga) şi destinația acestora (coloana din dreapta):

(1) Telnet;
(2) FTP;
(3) E-mail;
(4) WWW.

(a) transferul fişierelor de pe calculatorul aflat la distanță pe calculatorul utilizatorului;
(b) căutarea informațiilor pe discul magnetic al calculatorului şi afişarea lor la ecran;
(c) editarea imaginilor grafice şi transmiterea lor în *Internet*;
(d) transmiterea scrisorilor electronice;
(e) prelucrarea datelor cu ajutorul unei aplicații de calcul tabelar şi transmiterea rezultatelor prin poşta electronică;
(f) prezentarea, transmiterea şi căutarea informațiilor în *Internet*;
(g) acces la alte calculatoare, calculatorul utilizatorului devenind un terminal al calculatorului aflat la distanță.

14. Indicați tehnologia de cooperare în rețeaua *Internet*:
a) de la egal-la-egal;
b) client-server;
c) de la egal-la-egal şi client-server.

15. Care dintre afirmațiile ce urmează sînt adevărate:
a) la crearea paginilor Web se foloseşte limbajul PASCAL;
b) pentru a trimite o scrisoare electronică, avem nevoie de o conexiune *Internet*;
c) o pagină Web poate conține numai text şi imagini;
d) la crearea paginilor Web se foloseşte limbajul HTML;
e) o pagină Web poate conține text, imagini, secvențe sonore şi secvențe video;
f) scrisorile electronice nu pot fi citite de persoanele străine;
g) serverul de căutare poate citi informațiile de pe orice calculator conectat la *Internet?*

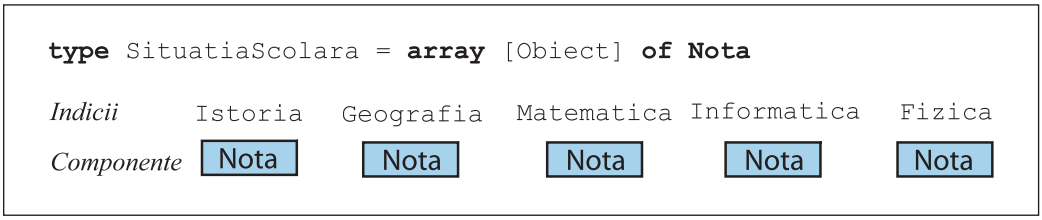
16. Selectați din lista ce urmează adresele de cutii poştale:

a) www.directie.orhei.md;
b) apetrescu@contabilitate.orhei.md;
c) ivulpe@presedinte.orhei.md;
d) bjosu@directie.cahul.md;
e) www.contabilitate.cahul.md;
f) munteanu@presedinte.calarasi.md.

RĂSPUNSURI LA TESTELE DE AUTOEVALUARE

Testul nr. 1

1.



2. Mulțimea de valori ale tipului de date `OrarulLectiilor` este constituită din tablouri bidimensionale. Liniile sînt specificate prin indici de tipul `ZiDeScoala`, iar coloanele – prin indici de tipul `Lectie`.

3.

a) `x[1]+ x[2]+ x[3]+ x[4];`

b) `y[7]+ y[8]+ y[9]+ y[10];`

c) `abs(x[3]);`

d) `abs(y[6]);`

e) `x[1]+ y[10].`

4.

```
Program RTA9;
{ Raspuns la Testul 1, Itemul 4 }
type Numere = array [1..50] of integer;
var A : Numere;
    n, i : integer;
begin
  write('Dati n='); readln(n);
  { Citim numerele de la tastatura }
  for i:=1 to n do
    begin
      write('A[', i, ']='); readln(A[i]);
    end;
  { Afisam numerele la ecran }
  writeln('Numerele in ordine inversa:');
```

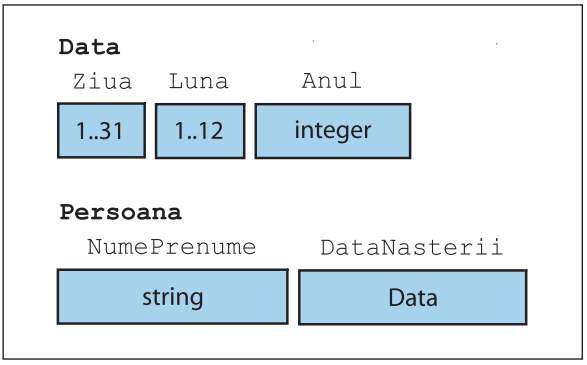
```
  for i:=n downto 1 do
    writeln('A[', i, ']=', A[i]);
  readln;
end.
```

5. Asupra şirurilor de tip **`string`** se poate efectua operația de concatenare (juxtapunere), notată prin „+”. 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`. De asemenea, asupra şirurilor de caractere sînt admise operațiile relaționale <, <=, =, >=, >, <>. Rezultatele acestor operații sînt de tipul `boolean`.

6.

```
Program RTA10;
{ Raspuns la Testul 1, Itemul 6 }
var S : string;
    i : integer;
begin
  writeln('Dati sirul de caractere:');
  readln(S);
  writeln('Sirul in ordine inversa:');
  for i:=length(S) downto 1 do
    write(S[i]);
    writeln;
  readln;
end.
```

7.



8.

```
Program RTA11;
{ Raspuns la Testul 1, Itemul 8 }
type Angajat = record
                 NumePrenume : string;
                 Salariu : real;
               end;
     ListaDePlata = array [1..100] of Angajat;
var L : ListaDePlata;
```

```
    s : real;
    i, n : integer;
begin
  write('Dati n='); readln(n);
  { Citirea datelor despre fiecare angajat }
  for i:=1 to n do
    begin
      writeln('Dati datele despre angajatul ', i);
      write('Numele si prenumele: ');
      readln(L[i].NumePrenume);
      write('Salariul: ');
      readln(L[i].Salariu);
    end;
  { Cautarea salariului maximal }
  s:=0;
  for i:=1 to n do
    if L[i].Salariu > s then s:=L[i].Salariu;
  writeln('Salariul maximal = ', s:10:2);
  { Afisarea informatiilor }
  writeln('Angajatii cu cel mai mare salariu:');
  for i:=1 to n do
    if L[i].Salariu = s then writeln(L[i].NumePrenume);
  readln;
end.
```

9. Instrucțiunea `with` se utilizează pentru a reduce textele programelor PASCAL prin excluderea repetărilor de denumiri ale variabilelor de tip **`record`** (articol).

10. Valorile variabilei `V`:

[ ], ['X'], ['Y'], ['Z'], ['X', 'Y'], ['X', 'Z'], ['Y', 'Z'], ['X', 'Y', 'Z'].

Valorile variabilei `I`:

[ ], [8], [9], [8, 9].

11.

```
Program RTA12;
{ Raspuns la Testul 1, Itemul 11 }
var S : string;
    i, n : integer;
begin
  write('Dati un sir de caractere format din ');
  writeln('literele mari ale alfabetului latin:');
  readln(S);
  { Numaram vocalele in sir }
  n:=0;
  for i:=1 to length(S) do
    if S[i] in ['A', 'E', 'I', 'O', 'U'] then n:=n+1;
  writeln('Numarul de vocale = ', n);
readln;
end.
```

12. Datele fişierelor PASCAL se păstrează pe suporturile de informaţie ale echipamentelor periferice: discuri şi benzi magnetice, discuri optice, hîrtia imprimantei sau a dispozitivului de citit documente ş.a. Procedura `assign` asociază variabilele de tip *fişier* din componenţa programelor PASCAL cu fişierele de pe suporturile de informaţie ale echipamentelor periferice.

13. În funcţie de **tipul operaţiilor permise** asupra componentelor, fişierele se clasifică în:
– fişiere de intrare (este permisă numai citirea);
– fişiere de ieşire (este permisă numai scrierea);
– fişiere de actualizare (sînt permise scrierea şi citirea).
În funcţie de **modul de acces** la componente, fişierele se clasifică în:
– fişiere cu acces secvenţial sau secvenţiale (accesul la componenta *i* este permis după ce s-a citit/scris componenta *i* – 1);
– fişiere cu acces aleator sau direct (orice componentă se poate referi direct prin numărul ei de ordine *i* în fişier).

14.

```
Program RTA13;
{ Raspuns la Testul 1, Itemul 14 }
type Angajat = record
                 NumePrenume : string;
                 Salariu : real;
               end;
     FisierAngajati = file of Angajat;
var A : Angajat;
    F : FisierAngajati;
    i, n : integer;
begin
  { Crearea fisierului SALARII.DAT }
  assign(F, 'SALARII.DAT');
  rewrite(F);
  write('Dati n='); readln(n);
  for i:= 1 to n do
    begin
      writeln('Dati datele angajatului ', i);
      write('Nume, prenume: '); readln(A.NumePrenume);
      write('Salariul: '); readln(A.Salariu);
      { Scrierea datelor despre angajat in fisierul F }
      write(F, A);
    end;
  close(F);
end.
```

15.

```
Program RTA14;
{ Raspuns la Testul 1, Itemul 15 }
type Angajat = record
                 NumePrenume : string;
                 Salariu : real;
               end;
     FisierAngajati = file of Angajat;
```

```
var A : Angajat;
    F : FisierAngajati;
begin
  assign(F, 'SALARII.DAT');
  reset(F);
  writeln('Datele citite din fisierul SALARII.DAT:');
  while not eof(F) do
    begin
      read(F, A);
      writeln(A.NumePrenume, ' ', A.Salariu:10:2);
    end;
  readln;
  close(F);
end.
```

16.

```
Program RTA15;
{ Raspuns la Testul 1, itemul 16 }
var a, b : real;
    c : string;
    Intrare, Iesire : text;
begin
  assign(Intrare, 'REZULTAT.TXT');
  reset(Intrare);
  assign(Iesire, 'MEDII.TXT');
  rewrite(Iesire);
  while not eof(Intrare) do
    begin
      readln(Intrare, a, b, c);
      writeln(Iesire, (a+b)/2, ' ', c);
      writeln((a+b)/2, ' ', c);
    end;
  readln;
  close(Intrare);
  close(Iesire);
end.
```

Testul nr. 2

1. 000, 001, 010, 011, 100, 101, 110, 111.

2. Rezultatul decodificării: CACB. Rezultatul codificării: 00001 00000 00010.

3. Alfabetul latin este format din 26 de litere. Prin urmare, numărul de mesaje posibile ale sursei de informație $n = 26 + 26 = 52$. Cantitatea de informație într-un mesaj:

$$I = \log_2 52 \approx 5{,}7 \text{ } biți.$$

Din inegalitatea $m \geq I$, obținem $m = 6$.

4. Numărul de mesaje posibile ale calendarului electronic $n = 31 \times 12 \times 100 = 37\,200$. Cantitatea de informație într-un mesaj:

$$I = \log_2 37\,200 \approx 15{,}2 \text{ } biți.$$

Din inegalitatea $m \geq I$, obținem $m = 16$.

5. $I =$ 8 *biți* (1 *octet*).

6. $V =$ 10 min. $\times$ 200 de caractere/minut $\times$ 1 octet = 2000 de *octeți*.

7.

```
Program RTA16;
 { Raspuns la Testul 2, Itemul 7 }
var c : char;
begin
  while not eof do
    begin
      readln(c);
      writeln(c, ' - ', ord(c));
    end;
end.
```

8.

```
Program RTA17;
 { Raspuns la Testul 2, Itemul 8 }
var i : integer;
begin
  while not eof do
    begin
      readln(i);
      writeln(i, ' - ', char(i));
    end;
end.
```

9. $I = (10 \times 10) \times 30 \times \log_2 128 = 21\,000$ de *biți* = 2625 de *octeți*.

10. *a)*, *c)*, *d)*, *f)*, *h)*.

11. $V = 3 \cdot I = 3 \cdot \log_2 \left(\frac{|42-34|}{0{,}1} + 1 \right) \approx 19{,}02 \text{ } biți.$

12. $V = \frac{30 \cdot 60}{3 \cdot 10^{-5}} \log_2 128 = 4{,}2 \cdot 10^8 \text{ } biți \approx 400{,}5 \text{ } Mbiți.$

13. (1) – (e); (2) – (f); (3) – (a); (4) – (c).

14. (1) – (c); (2) – (f); (3) – (a); (4) – (e).

Testul nr. 3

1. *a)* 667; *b)* ≈31,5926; *c)* ≈176,7551.

2. *b)* 8 nu este o cifră octală; *c)* 2, 8, 4 şi 6 nu sînt cifre binare.

3.

```
Program RTA18;
{ Raspuns la Testul 3, Itemul 3 }
var b : integer;  { baza }
    x : string;   { numarul citit de la tastatura }
    y : integer;  { numarul transformat in baza 10 }
    i : integer;
    bi : integer; { baza la puterea i }
begin
  write('Dati baza b='); readln(b);
  write('Dati un numar scris in baza ', b, ': ');
  readln(x);
  y:=0;
  bi:=1; { baza la puterea 0 }
  for i:=length(x) downto 1 do
    begin
      y:=y+(ord(x[i])-48)*bi;
      bi:=bi*b;
    end;
  writeln('Numarul scris in baza 10: ', y);
  readln;
end.
```

4. *a)* ≈1101,1110001; *b)* 111011,10110001; *c)* 10101000,011101001111.

5. *a)* ≈23036,7341; *b)* 1333,272; *c)* 155206,542.

6. *a)* ≈24FF,D611; *b)* 2DDB,534; *c)* 115,7F.

7. $(222221)_4$, $(1000001111)_2$, $(BB)_{16}$, $(132)_8$.

8. $(1D3)_{16} = (723)_8$; $(25)_8 = (21)_{10}$.

9.

```
Program RTA19;
{ Raspuns la Testul 3, Itemul 9 }
var x : string;  { numarul octal }
    y : string;  { numarul binar }
    i : integer;
    T : array ['0'..'7'] of string;
    { T[i] - reprezentarea binara a cifrei octale i }
begin
  { Initializarea tabelului }
  T['0']:='000'; T['1']:='001';
  T['2']:='010'; T['3']:='011';
  T['4']:='100'; T['5']:='101';
```

```
    T['6']:='110';  T['7']:='111';
  write('Dati un numar octal x='); readln(x);
  y:='';
  for i:=1 to length(x) do
    y:=y+T[x[i]];
  writeln('Reprezentarea binara: ', y);
  readln;
end.
```

10. *a)* 1001000111; *b)* 111100001; *c)* 100100111; *d)* 111010.

11.

0	1	1	1	1	1	0	1

12.

1	0	1	0	0	1	0	1

13. *a)* –1; *b)* +64; *c)* –39.

14.

1	1	1	0	0	0	0	0

15. *a)* $0{,}10101001\times 2^{5}$; *b)* $-0{,}100100101\times 2^{7}$; *c)* $-0{,}11\times 2^{-3}$.

16. $b = 8$; $x = 25$.

17. *a)* –26; *b)* 45; *c)* –8.

18. *a)* [–8, 7]; *b)* [–128, 127]; *c)* [–32 768, 32 767].

19. *a)* –0,375; *b)* 0,125; *c)* –0,5.

20. $A = -0{,}9921875$; $B = 0{,}9921875$.

21. *a)* –0,109375; *b)* 7,5; *c)* –0,25.

22.

1	0	1	0	1	0	1	1

23. $A = -7{,}5$; $B = 7{,}5$.

Testul nr. 4

1.

x	y	z	$\bar{z}$	$y\bar{z}$	$x \vee y\bar{z}$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	0	1

x	y	z	$\bar{z}$	$y\bar{z}$	$x \vee y\bar{z}$
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	0	1

2. *a)* şi *d)*.

3. {000, 010, 100, 110, 111}.

4.

```
Program RTA20;
{ Raspuns la Testul 4, Itemul 4 }
var x, y, z, f : boolean;
    xx, yy, zz, ff : 0..1;
begin
  write('Dati x='); readln(xx);
  if xx=0 then x:=false else x:=true;
  write('Dati y='); readln(yy);
  if yy=0 then y:=false else y:=true;
  write('Dati z='); readln(zz);
  if zz=0 then z:=false else z:=true;
  f:=x and (not y) or z;
  if f=false then ff:=0 else ff:=1;
  writeln('Valoarea expresiei logice: ', ff);
  readln;
end.
```

5.

```
Program RTA21;
{ Raspuns la Testul 4, Itemul 5 }
var x, y, z, f : boolean;
    xx, yy, zz, ff : 0..1;
begin
  writeln('x y z  f');
  writeln('———');
  for x:=false to true do
    for y:=false to true do
      for z:=false to true do
        begin
          f:=(not x) and (not y) or z;
          if x=false then xx:=0 else xx:=1;
          if y=false then yy:=0 else yy:=1;
          if z=false then zz:=0 else zz:=1;
          if f=false then ff:=0 else ff:=1;
          writeln(xx, ' ', yy, ' ', zz, '  ', ff);
        end;
  readln;
end.
```

6. Domeniul de definiție: {000, 001, 010, 011, 100, 101, 110, 111}. Domeniul de valori: {0, 1}.

7.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

8.

```
Program RTA22;
{ Raspuns la Testul 4, Itemul 8 }
var x1, x2,x3, y : boolean;
    xx1, xx2, xx3, yy : 0..1;
begin
  write('Dati x1='); readln(xx1);
  if xx1=0 then x1:=false else x1:=true;
  write('Dati x2='); readln(xx2);
  if xx2=0 then x2:=false else x2:=true;
  write('Dati x3='); readln(xx3);
  if xx3=0 then x3:=false else x3:=true;
  y:=x1 and ((not x2) or (not x3));
  if y=false then yy:=0 else yy:=1;
  writeln('Valoarea functiei logice y=', yy);
  readln;
end.
```

9.

```
Program RTA21;
{ Raspuns la Testul 4, Itemul 9 }
var x1,x2, x3, y : boolean;
    xx1, xx2, xx3, yy : 0..1;
begin
  writeln('x1 x2 x3  y');
  writeln('———————');
  for x1:=false to true do
    for x2:=false to true do
      for x3:=false to true do
        begin
          y:=(not x1) and (x2 or x3);
          if x1=false then xx1:=0 else xx1:=1;
          if x2=false then xx2:=0 else xx2:=1;
          if x3=false then xx3:=0 else xx3:=1;
```

```
            if y=false then yy:=0 else yy:=1;
          writeln(' ', xx1, '  ', xx2, '  ', xx3, '  ', yy);
        end;
  readln;
end.
```

10. $2^{2^5} = 4294967296$.

11.

ŞI-NU

x_1	x_2	$\overline{x_1 x_2}$
0	0	1
0	1	1
1	0	1
1	1	0

SAU-NU

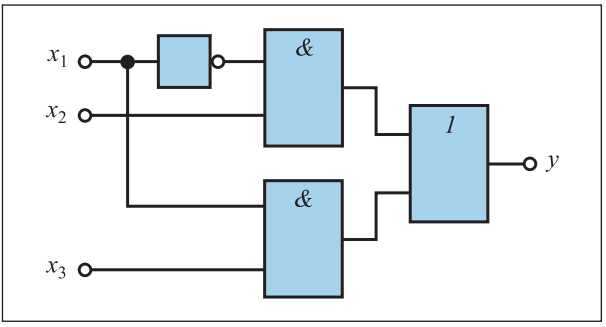
x_1	x_2	$\overline{x_1 \vee x_2}$
0	0	1
0	1	0
1	0	0
1	1	0

12. *a)* şi *c)*; *b)* şi *d)*.

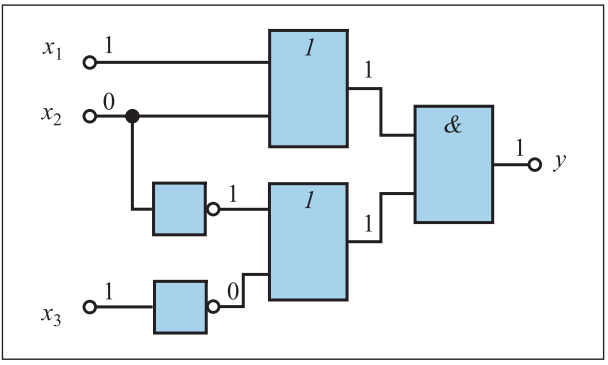
13. *b)*.

Testul nr. 5

1.



2.



3. $y = (x_1 \vee x_2)\,(\bar{x}_2 \vee \bar{x}_3)$.

4.

```
Program RTA24;
{ Raspuns la Testul 5, Itemul 4 }
var a, b, s, t : char;
begin
  write('Dati a='); readln(a);
  write('Dati b='); readln(b);
  if (a='0') and (b='0') then begin s:='0'; t:='0'; end;
  if (a='0') and (b='1') then begin s:='1'; t:='0'; end;
  if (a='1') and (b='0') then begin s:='1'; t:='0'; end;
  if (a='1') and (b='1') then begin s:='1'; t:='1'; end;
  writeln('s=', s, '  ', 't=', t);
  readln;
end.
```

5. Un semisumator conține 2 porți logice *NU*, 3 porți logice *ŞI* şi o poartă logică *SAU* (*fig. 5.10*). Un sumator elementar conține 2 semisumatoare şi o poartă logică *SAU* (*fig. 5.11*). Prin urmare, un sumator elementar conține 4 porți logice *NU*, 6 porți logice *ŞI* şi 3 porți logice *SAU*. Evident, un sumator de *n* biți conține 4*n* porți logice NU, 6*n* porți logice *ŞI* şi 3*n* porți logice *SAU* (*fig. 5.12*). Pentru un sumator de *n* = 8 biți, obținem: 32 de porți logice *NU*; 48 de porți logice *ŞI*; 24 de porți logice *SAU*.

6.

```
Program RTA25;
{ Raspuns la Testul 5, Itemul 6 }
label 1;
var A, B, S : string;
    t : char;   { transportul }
    i : integer;
begin
  writeln('Dati doua numere binare A si B. Fiecare numar');
  writeln('trebuie sa contina exact 8 cifre binare.');
  write('A='); readln(A);
  write('B='); readln(B);
  t:='0';        { initial transportul t=0 }
  S:='xxxxxxxx'; { initial suma S este necunoscuta }
  for i:=8 downto 1 do
    begin
       { adunarea cifrelor binare A[i], B[i], T[i-1] }
       if (A[i]='0') and (B[i]='0') and (t='0') then
         begin S[i]:='0'; t:='0'; goto 1; end;
       if (A[i]='0') and (B[i]='0') and (t='1') then
         begin S[i]:='1'; t:='0'; goto 1; end;
       if (A[i]='0') and (B[i]='1') and (t='0') then
         begin S[i]:='1'; t:='0'; goto 1; end;
       if (A[i]='0') and (B[i]='1') and (t='1') then
         begin S[i]:='0'; t:='1';  goto 1; end;
```

```
          if (A[i]='1') and (B[i]='0') and (t='0') then
            begin S[i]:='1'; t:='0'; end;
        if (A[i]='1') and (B[i]='0') and (t='1') then
          begin S[i]:='0'; t:='1'; goto 1; end;
        if (A[i]='1') and (B[i]='1') and (t='0') then
          begin S[i]:='0'; t:='1'; goto 1; end;
        if (A[i]='1') and (B[i]='1') and (t='1') then
          begin S[i]:='1'; t:='1'; goto 1; end;
  1: end;
  writeln('S=', S);
  writeln('Cifra de depasire t=', t);
  readln;
end.
```

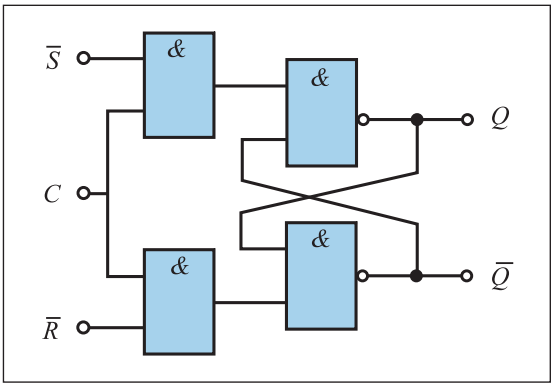
7. (1) – (e); (2) – (g); (3) – (a); (4) – (d); (5) – (h); (6) – (c).

8. y_1 – SUS; y_2 – JOS; y_3 – STÎNGA; y_4 – DREAPTA.

x_1	x_2	y_1	y_2	y_3	y_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

9. $Q = 1$ şi $\bar{Q} = 0$.

10.



11. (1) – (d); (2) – (i); (3) – (g); (4) – (a); (5) – (c); (6) – (f).

12. 00010011.

13. 11010000.

14.

```
Program RTA26;
{ Raspuns la Testul 5, Itemul 14 }
var R : string;
    n, m, i, j : integer;
```

```
begin
  writeln('Dati continutul initial al registrului');
  write('R='); readln(R);
  n:=length(R); { capacitatea registrului }
  write('Dati m='); readln(m);
  for j:=1 to m do
    begin
      { deplasarea cu o pozitie }
      for i:=1 to n-1 do R[i]:=R[i+1];
      R[n]:='0';
    end;
  writeln('Continutul registrului dupa deplasare');
  writeln('R=', R);
  readln;
end.
```

15. 10010011.

16. 10101001.

17.

```
Program RTA27;
{ Raspuns la Testul 5, Itemul 17 }
label 1;
var A : string;
    n, m, i, j : integer;
    t : char; { transportul }
begin
  writeln('Dati continutul initial al numaratorului');
  write('A='); readln(A);
  n:=length(A); { capacitatea numaratorului }
  write('Dati m='); readln(m);
  for j:=1 to m do
    begin
      { adunarea unitatii la continutul numaratorului }
      t:='1';
      for i:=n downto 1 do
        begin
          { adunarea transportului la cifra A[i] }
          if (A[i]='0') and (t='0') then
            begin A[i]:='0'; t:='0'; goto 1; end;
          if (A[i]='0') and (t='1') then
            begin A[i]:='1'; t:='0'; goto 1; end;
          if (A[i]='1') and (t='0') then
            begin A[i]:='1'; t:='0'; goto 1; end;
          if (A[i]='1') and (t='1') then
            begin A[i]:='0'; t:='1'; goto 1; end;
     1: end;
    end;
  writeln('B=', A);
  readln;
end.
```

Testul nr. 6

1. (1) – (i); (2) – (f); (3) – (a); (4) – (c); (5) – (d); (6) – (g).

2. Tastatura, mouse-ul, vizualizatorul, imprimanta, unitatea de disc magnetic, unitatea de disc optic, cititorul de documente (scanerul).

3. *a)*, *c)*, *f)*.

4. Magistrala asigură comunicarea procesorului cu memoria internă şi echipamentul periferic.

5. Vezi *figura 6.2*. Componentele obligatorii: procesorul, magistrala, memoria internă, controlerul, dispozitivul de intrare-ieşire.

6. *Codul instrucțiunii* – în acest cîmp se indică operația ce trebuie executată. *Adresă operand 1* şi *Adresă operand 2* – în aceste cîmpuri se indică adresele locațiilor din memoria internă care conțin, respectiv, primul şi al doilea operand. *Adresă rezultat* – indică adresa locației din memoria internă în care va fi depus rezultatul operației.

7. *a)* numărul din locația 101 este adunat cu numărul din locația 153, iar rezultatul obținut este depus în locația 342;

b) numărul din locația 508 este împărțit la numărul din locația 391, iar rezultatul obținut este depus în locația 216;

c) numărul din locația 751 este înmulțit cu numărul din locația 852, iar rezultatul obținut este depus în locația 031;

d) din numărul din locația 450 se scade numărul din locația 709, iar rezultatul obținut este depus în locația 011.

8. *a)*, *c)*, *f)* – instrucțiuni operaționale; *g)* – instrucțiune de transfer; *e)* – instrucțiune de salt; *b)*, *d)* – instrucțiuni de intrare-ieşire.

9.

```
INC X
MEM S
SCD Y
ADU X
```

10.

```
01 583
02 461
01 971
02 583
01 461
```

11. *a)*, *d)*, *f)*, *h)* – resurse tehnice; *b)*, *c)*, *e)*, *g)* – resurse programate.

12. În cazul memoriilor externe cu acces secvențial scrierea/citirea înregistrării dorite este posibilă doar după scrierea/citirea tuturor înregistrărilor precedente. În cazul memoriilor externe cu acces direct orice înregistrare poate fi scrisă/citită fără a mai citi/scrie înregistrările precedente.

13. *b)*, *c)*, *e)*.

14. *b), e)*.

15. 2867,2 *Kocteți/s*.

16. *a)*, *c)*, *d)*.

17. *a)*, *b)*, *e)*.

18. Parametrii de bază ai unui calculator:
– viteza de operare;
– capacitatea memoriei interne;
– componența, capacitatea şi timpul de acces ale unităților de memorie externă;
– componența şi parametrii tehnici respectivi ai echipamentelor periferice;
– parametrii de masă şi gabarit;
– costul.

19. Parametrii de bază ai microprocesorului:
– lungimea cuvîntului;
– frecvența ceasului de sistem;
– capacitatea magistralelor;
– capacitatea de prelucrare.

Testul nr. 7

1. *a)* ≈6 ore; *b)* 12 min. 30 s; *c)* ≈0,73 s; *d)* ≈0,00072 s.

2. *b), d)*.

3. Calculatoarele, adaptoarele de rețea, structura de comunicare.

4. *a)*, *c)*.

5. *a)*, *d)*.

6. Un protocol de rețea defineşte modul de adresare a calculatoarelor, lungimea şi componența pachetelor de date, algoritmul de depistare şi corectare a erorilor, modul de conectare fizică a adaptoarelor şi cablurilor de rețea.

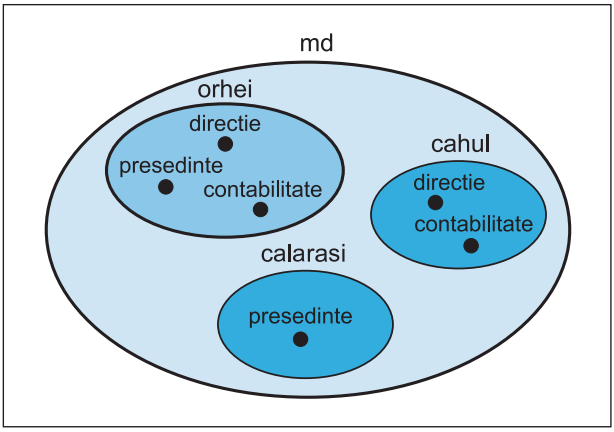
7. Arhitectura unei rețele reprezintă modul în care este concepută rețeaua: topologia, protocoalele de comunicație, tehnologiile de cooperare în rețea.

8. Vezi *figura 7.5*.

9. *d)*.

10. În *Internet* calculatoarele se identifică cu ajutorul adreselor. Acestea pot fi de două tipuri: adrese numerice şi adrese simbolice.

11. Întrucît $(214)_{10} = (11010110)_2$, iar biții 0-2 au valoarea 110, adresa *Internet* aparține clasei *C*. În cazul clasei *C* adresa subrețelei se indică în biții 3-32 ai adresei numerice. Din primul octet al adresei numerice selectăm biții 3-7 şi transformăm numărul binar obținut în sistemul zecimal: $(10110)_2 = (22)_{10}$. Prin urmare, adresa subrețelei este 22.121.216. Numărul 109 din ultimul octet al adresei numerice reprezintă adresa calculatorului în subrețea.



12. md – domeniul de cel mai înalt nivel. Acest domeniu include subdomeniile **orhei**, **cahul** şi **calarasi**. Subdomeniul **orhei** include calculatoarele **directie**, **contabilitate** şi **presedinte**. Subdomeniul **cahul** include calculatoarele **directie** şi **contabilitate**. Subdomeniul **calarasi** include calculatorul **presedinte**.

13. (1) – (g); (2) – (a); (3) – (d); (4) – (f).

14. *b)*.

15. *b)*, *d)*, *e)*.

16. *b)*, *c)*, *d)*, *f)*.

Bibliografie

1. Bolun Ion. *Inițiere în rețele. INTERNET*. Chișinău, Editura ASEM, 1997.

2. Ceapâru Mihai. *Comunicația prin intermediul rețelelor de calculatoare*. Bucureşti, Editura Tehnică, 1996.

3. Cerchez Emanuela. *Internet. Manual pentru liceu. Filiera teoretică*. Iaşi, Editura Polirom, 2000.

4. Cerchez Emanuela, Şerban Marinel. *Informatica pentru gimnaziu*. Iaşi, Editura Polirom, 2002.

5. Gremalschi Anatol, Mocanu Iurie, Spinei Ion. *Informatica. Limbajul PASCAL*. Chişinău, Editura Ştiința, 2003.

6. Gremalschi Ludmila, Mocanu Iurie. *Structura şi funcționarea calculatorului*. *Material didactic pentru licee şi colegii*. Chişinău, Editura Lyceum, 1996.

7. Gremalschi Anatol, Bejan Viorel, Gremalschi Ludmila. *Structura calculatoarelor numerice*. *Material didactic*. Chişinău, UTM, 1996.

8. Levine John R. *Internet pentru toți*. Bucureşti, Editura Teora, 1996.

9. Lowe Doug. *Rețele pentru toți*. Bucureşti, Editura Teora, 1995.

10. Mârşanu Radu. *Sisteme de calcul. Manual pentru licee de informatică, clasa a IX-a*. R.A. Bucureşti, Editura Didactică şi Pedagogică, 1995.

11. Mârşanu Radu, Velicanu Manole. *Tehnică de calcul. Manual pentru clasa a XII-a*. Bucureşti, Editura ALL, 1999.

12. Mihoc Dan, Iliescu Sergiu Stelian. *Elemente de informatică. Mecanizarea şi automatizarea producției. Manual pentru licee industriale, clasa a XII-a*. R.A. Bucureşti, Editura Didactică şi Pedagogică, 1995.

13. Mucenic Băşoiu, Mihai Băşoiu, Eugen Ştefan. *Compact disc*. Bucureşti, Editura Teora, 1995.

14. Pfaffenberger Bryan, Petersen Judy. *Dicționar explicativ de calculatoare*. Bucureşti, Editura Teora, 1996.

15. Petrescu Adrian, Iacob Francisc, Racoviță Zoe. *Inițiere în structura calculatoarelor electronice*. Bucureşti, Editura Teora, 1996.

16. Petrescu Silviu. *Informatica aplicată: manual de informatică pentru clasa a XII-a*. Bucureşti, Editura Teora, 1998.

17. Secrieru Nicolae, Gremalschi Anatol, Cornea Ion. *Arhitectura şi organizarea microprocesoarelor*. Chişinău, Editura Universitas, 1995.

18. Velicanu Manole, Vasilescu Adrian. *Bazele informaticii. Manual pentru clasa a XII-a*. Bucureşti, Editura ALL, 1999.

19. Залогова Л.А., Плаксин М.А., Русаков С.В., Русакова О.Л. и др. *Информатика. Задачник-практикум в 2 т.* / Под ред. Семакина И.Г., Хеннера Е.К.: Том 1. – М.: Лаборатория Базовых Знаний, 1999.

20. Семакин И.Г., Залогова Л.А., Русаков С.В., Шестакова Л.В. *Информатика. Базовый курс для 7-9 классов*. – М.: Лаборатория Базовых Знаний, 1999.

Manualul acesta este proprietatea Ministerului Educației al Republicii Moldova.

Liceul _____

Manualul nr. _____

Anul de folosire	Numele şi prenumele elevului	Anul şcolar	Aspectul manualului	
			la primire	la restituire
1.				
2.				
3.				
4.				
5.				

• Dirigintele trebuie să controleze dacă numele elevului este scris corect.

• Elevul nu trebuie să facă niciun fel de însemnări pe pagini.

• Aspectul manualului (la primire şi la restituire) se va aprecia folosind termenii: *nou, bun, satisfăcător, nesatisfăcător*.

Imprimare la Tipografia „SEREBIA” SRL
str. Calea Ieşilor, 10;
MD-2069, Republica Moldova
Comanda nr. 105