Capitolul 1

STRUCTURI DE DATE

1. ORGANIZAREA DATELOR

STUDIU DE CAZ ***Compania Eficient***

Pentru reducerea cheltuielilor cu amenajarea spațiilor comerciale, reclamă și personal, compania Eficient selectează tineri distribuitori, absolvenți de liceu. Oferta companiei este foarte atrăgătoare; de aceea, numărul solicitanților depășește numărul de posturi oferite (n). Selecția candidaților se face în ordinea înregistrării scrisorii de intenție și a CV-ului. Dosarele candidaților sunt păstrate într-un fișet, unul peste altul, în ordinea angajării. Periodic, compania trimite un angajat la cursuri de formare; este trimis întotdeauna, ultimul angajat (fig. 1). Angajații sunt plătiți în funcție de numărul de produse distribuite (vândute) zilnic. Săptămânal, managerul companiei ține evidența pe zile a produselor distribuite de fiecare angajat. În orice moment, managerul poate determina angajatul cu cea mai bună activitate într-o zi sau ziua în care a fost distribuit cel mai mic număr de produse. La sfârșitul săptămânii, după ce aplică bonusuri sau penalizări, managerul centralizează aceste date într-un registru de evidență a vânzărilor realizate de către fiecare angajat.

Managerul companiei dorește să prelucreze cu calculatorul datele pentru selecția candidaților, evidența angajaților și evidența vânzărilor.

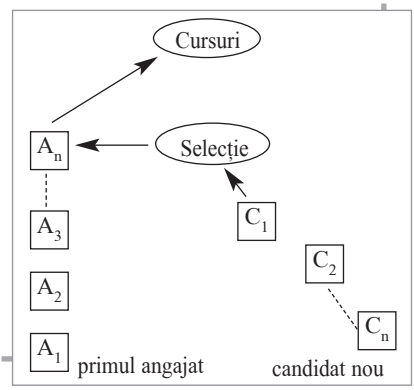


Figura 1

1.1. Analiza problemei

☑ datele despre candidați; din CV-ul fiecărui candidat vor fi reținute următoarele informații:

– numele,
– anul nașterii,
– media la examenul de bacalaureat;

	L	Ma	Mi	J	V	S	D
A_1							
A_2							
A_n							

Evidența săptămânală

Figura 2

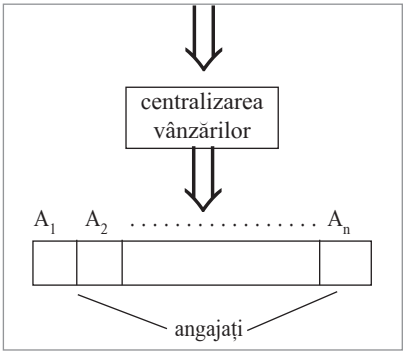


Figura 3

☑ selecția candidaților se face în ordinea înregistrării datelor personale;

☑ după examinarea unui candidat (selecție), datele acestuia nu mai sunt necesare; va intra în selecție candidatul următor;

☑ întotdeauna, datele unui nou candidat sunt așezate după datele ultimului candidat care așteaptă pentru selecție;

☑ datele angajaților sunt păstrate într-o ordine care să permită numirea rapidă a ultimului angajat în vederea trimiterii sale la cursuri;

☑ pentru evidența săptămânală a vânzărilor, se reține numărul de produse distribuite (vândute) zilnic de către fiecare angajat (fig. 2);

☑ pentru evidența centralizată a vânzărilor, totalul vânzărilor realizate într-o săptămână de fiecare angajat se adaugă la vânzările realizate de acel agajat până la momentul respectiv (fig. 3).

1.2. Soluția problemei

☑ problema propusă de managerul companiei Eficient necesită prelucrări matematice cu un grad mic de dificultate: centralizarea săptămânală, prin însumare, a valorilor reprezentând vânzările realizate de către fiecare angajat;

☑ datele specifice problemei trebuie organizate avantajos, astfel încât folosirea calculatorului să înlocuiască fișetele sau mapele în care sunt păstrate dosarele candidaților/angajaților și să respecte cerințele de așteptare sau prioritate specifice problemei.

1.3. Organizarea datelor

- Din analiza problemei urmărind *semnificația și complexitatea datelor*, rezultă două categorii de date:

☒ *date elementare*, spre exemplu: numărul de produse vândute de un angajat la un moment dat;

☒ *date grupate (structurate):*

– *date cu aceeași semnificație* (de același tip) – numite și *date omogene* –, spre exemplu: evidența săptămânală a vânzărilor pe zile și angajați, evidența vânzărilor pe angajați, datele despre toți angajații/candidații;

– *date cu semnificații diferite* – numite și *date neomegene* – spre exemplu, datele prin care este descrisă o persoană (candidat sau angajat): nume, an, medie.

De reținut!

O persoană este descrisă prin mai multe tipuri de date, ceea ce determină aspectul neomogen al grupului de date.

Candidații sau angajații formează grupuri de același tip – *persoană* – de unde rezultă aspectul omogen al acestui grup de date.

- Din analiza problemei urmărind *restricțiile de intrare-ieșire* (așteptare sau prioritate), după care sunt organizate datele dintr-un grup (structură), rezultă două categorii de date:

☒ date organizate după disciplina specifică unei *cozi* sau *fir de așteptare*; spre exemplu, candidații care așteaptă pentru selecție. Într-o astfel de structură, întotdeauna, un element nou este așezat (intră) după ultimul element. Dintr-o astfel de structură, iese, întotdeauna, primul element – în exemplul nostru, candidatul aflat „la rând".

☒ date organizate după disciplina specifică așezării obiectelor unul peste altul, în *stivă*; spre exemplu, dosarele angajaților. Într-o astfel de structură, întotdeauna, un element nou este așezat deasupra celorlalte, în *vârful* stivei. Dintr-o astfel de structură, iese, întotdeauna, elementul din *vârful* stivei – în exemplul nostru, va fi prelucrat dosarul ultimului angajat.

Concluzie

- organizarea datelor specifice unei probleme se poate face în locații de memorie independente – *date elementare* – sau în locații grupate – *structuri de date;*
- structurile de date pot fi:
  – *structuri omogene* (date cu aceeași semnificație/tip),
  – *structuri neomegene* (date cu semnificații/tipuri diferite);
- structurile omogene se numesc *tablouri;*
- într-un tablou, datele pot fi organizate astfel:

– după un singur criteriu – *tablouri unidimensionale* sau *vectori*; spre exemplu, tabloul pentru evidența vânzărilor realizate de fiecare angajat,

– după două criterii – *tablouri bidimensionale* sau *matrice*; spre exemplu, tabloul pentru evidența vânzărilor realizate zilnic (criteriul 1 – zilele săptămânii) de către fiecare angajat (criteriul 2 – angajații);

- într-un tablou unidimensional, datele pot fi organizate astfel încât să respecte o disciplină de intrare/ieșire de tip *coadă* sau *stivă*;
- structurile neomogene se numesc *articole* sau *înregistrări*; mai multe articole care descriu aceeași entitate (obiect, persoană etc.) reprezintă un grup de date cu aceeași semnificație și poate fi organizat într-o structură omogenă de tip tablou unidimensional (*vector de articole*).

TEME

1. Explicați deosebirea dintre analiza datelor din punct de vedere al complexității și analiza datelor din punct de vedere al regulilor de organizare (intrare/ieșire) într-o structură (grup de date).

2. Formulați un exemplu care să evidențieze diferența dintre datele omogene și datele neomegene.

3. Formulați un exemplu care să necesite organizarea după mai multe criterii a datelor cu aceeași semnificație.

4. Formulați un exemplu care să necesite organizarea datelor cu aceeași semnificație în structuri de tip *coadă*.

5. Formulați un exemplu care să necesite organizarea datelor cu aceeași semnificație în structuri de tip *stivă*.

6. Explicați disciplina structurii de date de tip coadă (FIFO – First Input First Output = *primul intrat, primul ieșit*).

7. Explicați disciplina structurii de date de tip stivă (LIFO – Last Input First Output = *ultimul intrat, primul ieșit).*

8. Formulați exemple de oganizare a datelor după alte reguli decât cele specifice structurilor de tip *coadă* sau *stivă*.

TEMĂ de GRUP

Densitatea straturilor geologice

*Un grup de geologi studiază densitatea straturilor geologice. În acest scop, s-a extras câte o probă pentru fiecare dintre cele **n** straturi geologice studiate. Fiecare probă este transmisă la un laborator specializat; pentru fiecare probă se fac mai multe teste, cel mult **m**. Întrucât nu există suficiente aparate, analiza de laborator durează. La sfârșitul activității, geologii păstrează probele în ordinea naturlă a straturilor geologice* (fig. 4). *Rezultatele testelor se păstrează astfel încât să se poată determina, cu ușurință:*

– zăcământul la care s-au obținut aceleași valori la toate testele;

*– zăcământul cu cea mai mare densitate medie la cele **m** teste;*

*– zăcământul la care s-a obținut cea mai mare diferență între densitatea maximă și densitatea minimă din cele **m** teste.*

Cerințe:

1. Identificați formele de organizare a datelor specifice problemei propuse.

2. Fiecare membru al grupului descrie una dintre formele de organizare a datelor identificate (justificare, necesitate, proprietăți, prelucrări specifice și alte aspecte).

3. Grupul compune un scenariu pentru prelucrarea cu calculatorul a datelor specifice acestei probleme; scenariul poate fi prezentat narativ sau organizat pe secvențe (pași).

4. Grupul realizează o prezentare PowerPoint care să ajute la susținerea temei.

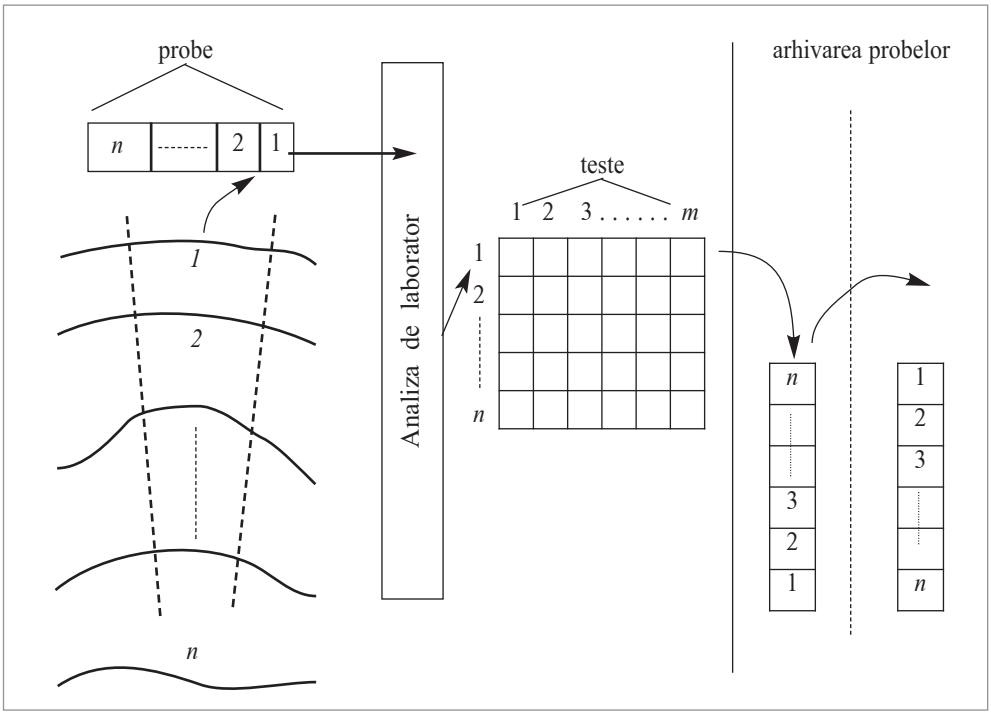


Figura 4

SUGESTIE DE PREZENTARE:

Prezentările pot fi expuse și analizate în laboratorul de Informatică; se va urmări corectitudinea rezolvării, creativitatea prezentării și eficiența lucrului în grup.

2. ORGANIZAREA DATELOR CU ACEEAȘI SEMNIFICAȚIE ÎN TABLOURI BIDIMENSIONALE

APLICAȚIA 1 FLOARE DE COLȚ

Membrii clubului „Floare de colț” participă la o tabără de vară, timp de o săptămână, într-o zonă montană greu accesibilă. Scopul lor este să înregistreze în fiecare zi temperatura aerului și să determine temperatura minimă, temperatura maximă și zilele în care s-au atins aceste temperaturi.

1. *Analiza problemei*

- *Date de intrare*
  – 7 valori reprezentând temperatura înregistrată în fiecare zi a săptămânii
- *Date de ieșire*
  – temperatura minimă (**tmin**);
  – temperatura maximă (**tmax**);
  – ziua/zilele în care s-au atins **tmin** și **tmax.**

2. *Organizarea datelor*

Datele de intrare au aceeași semnificație și vor fi înregistrate într-un vector (**T**) cu 7 elemente.

De exemplu:

T	18	15	12	14	18	15	12

tmin = 12 înregistrată miercuri și duminică
tmax = 18 înregistrată luni și vineri

3. *Raționamentul problemei*

Se înregistrează temperaturile zilnice în vectorul **T**. Se determină, printr-o singură parcurgere, valoarea minimă și cea maximă.

Pentru afișarea zilei/zilelor se mai fac două parcurgeri ale vectorului. Ziua va fi afișată ca indice (1, 2 etc.). Pentru afișarea zilei prin nume (luni, marți etc.) se recomandă introducerea structurii selective după **zi**.

4. *Reprezentarea algoritmului*

```
begin temperaturi1
alocă T[7]
pentru zi = 1, 7 execută citește T[zi] sfârșit pentru
tmin ← T[1]
tmax ← T[1]
pentru zi = 2, 7 execută
    bloc
        dacă T[zi] < tmin atunci tmin ← T[zi]
        sfârșit dacă
        dacă T[zi] > tmax atunci tmax ← T[zi]
        sfârșit dacă
    sfârșit bloc
```

```
sfârșit pentru
        scrie tmin
        pentru zi = 1, 7 execută
                dacă T[zi] = tmin atunci scrie zi
                sfârșit dacă
        sfârșit pentru
        scrie tmax
        pentru zi = 1, 7 execută
                dacă T[zi] = tmax atunci scrie zi
                sf dacă
        sfârșit pentru
        sfârșit temperaturi1
```

APLICAȚIA 2 FLOARE DE COLȚ

Măsurarea temperaturii aerului în zonele greu accesibile a fost foarte apreciată de ecologiști. Numărul voluntarilor dornici să participe la astfel de acțiuni a crescut. Instructorul clubului a organizat zece echipe care să măsoare temperatura zilnică în diverse zone de interes.

Prelucrarea valorilor înregistrate se va face astfel:

– se afișează temperatura minimă și temperatura maximă înregistrate în cele zece zone pe parcursul săptămânii;

– se afișează temperatura medie, pentru oricare dintre zone, la solicitarea celui interesat;

– se afișează zona în care s-a atins cea mai mică, respectiv cea mai mare temperatură, pentru orice zi a săptămânii solicitată de cel interesat.

1. *Analiza problemei*

Datele problemei au aceeași semnificație – temperatura zilnică –, dar se referă la zone diferite; de aceea, pentru organizarea lor se introduce și criteriul **zonă**. Rezultă un tablou (**T**) cu două dimensini: **zona** (linie) și **ziua** (coloană) (fig. 5).

T

Zona							
1						21	
2						18	
3	18	15	12	14	18	15	12
4						16	
5						17	
6						19	
7						20	
8						20	
9						17	
10						18	
	1	2	3	4	5	6	7

Ziua

Figura 5

Cum organizăm datele în tablouri

1. Dacă într-o problemă este necesară memorarea mai multor valori cu aceeaşi semnificaţie, aceste valori vor fi grupate într-un ansamblu de tip **tablou**.
2. În funcţie de cerinţele problemei şi de semnificaţia datelor, acestea sunt organizate în tablouri cu o singură dimensiune, numite **vectori**, sau în tablouri cu două dimensiuni, numite **matrice**.
3. Elementele unui tablou se identifică printr-o adresă formată din numele tabloului şi câte un indice pentru fiecare dimensiune.
4. La tablourile cu două dimensiuni, primul indice din adresă reprezintă linia, iar cel de-al doilea coloana tabloului.
5. Operaţiile care se repetă pentru fiecare element din tablou pot fi grupate în structuri repetitive cu contor. Contorul generează chiar indicele de adresă.

3. IMPLEMENTAREA TABLOURILOR BIDIMENSIONALE

Pentru memorarea şi prelucrarea datelor organizate ca tablouri bidimensionale, înainte de scrierea unui program trebuie să cunoaştem următoarele elemente:

– tipul elementelor din tablou (datele cu aceeaşi semnificaţie),

– numărul maxim de elemente din tablou (capacitatea tabloului).

capacitatea tabloului = nrmax _linii * nrmax _coloane

Aceste elemente rezultă din analiza problemei şi sunt folosite de compilator pentru determinarea zonei de memorie alocată tabloului.

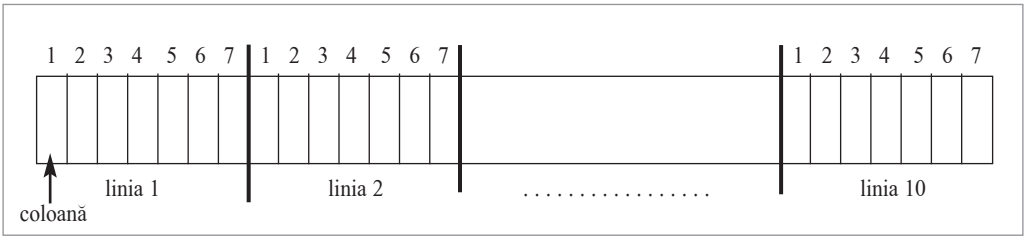


Figura 6 – Liniarizarea tabloului bidimensional

Tabloul ocupă în memoria calculatorului o „suprafaţă” (*array*) compusă din locaţii învecinate (adiacente). În fiecare locaţie este memorată valoarea unui element. Adresa locaţiilor începe de la o *valoare de referinţă* specifică limbajului de programare (corespunzătoare primei linii) şi se construieşte prin valori succesive (corespunzătoare coloanelor de pe linie), pentru fiecare element din tablou. În memoria calculatorului, tabloul este liniarizat: elementele sunt memorate în locaţii adiacente, în ordinea liniilor (fig. 6). Adresarea unui element se face printr-o pereche de indici corespunzători liniei şi coloanei pe care se află elementul respectiv.

În consecință, tipul variabilelor folosite ca indice de adresă trebuie să accepte valori în domeniile:

(1) [*valoare de referință, nrmax-linii*] pentru indicele de linie și

(2) [*valoare de referință, nrmax-coloane*] pentru indicele de coloană.

Adresarea elementelor se face, cel mai frecvent, prin două structuri repetitive cu contor, imbricate:

pentru indice_linie = valoare-de referință **la** nr-linii **execută**
  pentru indice_coloana = valoare-de referință **la** nr-coloane **execută**
    // prelucrarea elementelor din tablou în ordinea așezării pe linii
  sfârșit pentru
sfârșit pentru

Pentru indicele de linie sau indicele de coloană, se pot folosi tipuri diferite de date, ceea ce permite o referire asemănătoare cu cea întâlnită la jocul de șah; spre exemplu: T[2][C] reprezintă elementul de pe tabla de șah aflat pe linia 2 în coloana C (fig. 7).

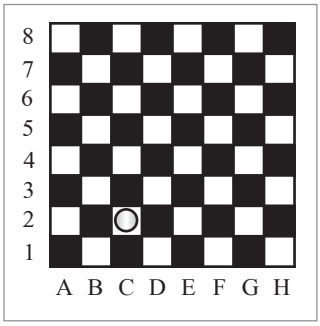


Figura 7

Elementele de sintaxă specifice tablourilor bidimensionale sunt prezentate în Tabelul 1.

***Tabelul* 1. Tablouri bidimensionale – elemente de sintaxă**

PASCAL	C / C++
Declararea tabloului	
var *tablou* **array** [1...*nl, 1..nc*] **of** *tipelement;*	*tipelement tablou*[*nl*][nc];
Declararea indicelui de adresă	
var *l,c:tipindice;*	*tipindice l,c;*
Adresarea unui element din tablou	
tablou [l,c] adresa de referință este 1.	*tablou* [*l*][*c*] adresa de referință este 0.
Exemplu: pentru tabloul *temperaturi* cu 100 de elemente de tip întreg distribuite pe 10 linii și 10 coloane: se inițializează cu zero elementele de pe primele 5 linii	

var *temperaturi* : **array** [1..10,1..10] **of integer**;
l,c: **byte;**
begin
for l:=1 to 5 do
for c:=1 to 10 do
temperaturi [l,c] =0;
end
	int *temperaturi* [10] [10];
	short l,c;
	{
	for (l=0; i< 5; i++)
	for (c=0; i< 10; i++)
	
	temperaturi [l][c]=0;
	}

Datele organizate în tablouri bidimensionale sunt prelucrate element cu element. Iată câteva dintre cele mai frecvente prelucrări:

- introducerea valorilor direct de la tastatură sau dintr-un fișier;
- afișarea valorilor pe ecran sau într-un fișier;
- verificarea unor proprietăți;
- compararea valorilor (aflarea elementului maxim sau minim);
- simularea unor situații reale.

Tabelul 2. **Operații la nivel de matrice**

PASCAL	C / C++
A. Introducerea valorilor în ordinea „pe linii”	

var A: array[1..10, 1..10] of integer;
l,c,m,n : byte;
begin
write (‘ numarul de linii=’); readln (m);
write (‘ numarul de coloane=’); readln (n);
for l:=1 to m do
  for c:=1 to n do
  readln (A[l,c]);
end;
	int A [10] [10] ;
	int l,c,m,n ;
	{
	cout<<” numarul de linii=” ; cin>>m;
	cout<<” numarul de coloane=” ; cin>>n
	for (l=0; l< m; l++)
	  for (c=0; c< n; c++)
	  cin >>A[l][c];
	}

B. Afișarea valorilor în ordinea „pe coloane” se păstrează declarațiile de la secvența A	

begin
for c:=1 to n do
 begin
  writeln;
  for l:=1 to m do
  write (A[l,c], ‘ ’);
  end;
end;
	for (c=0; c< n; c++)
	{
	  cout<<endl;
	  for (l=0; l< m; l++)
	  cout <<A[l][c]<< “ ”;
	}

C. Determinarea elementului minim de pe o linie oarecare, k se păstrează declarațiile de la secvența A	

var k:byte; min: integer;
begin
  write (‘specificati linia=’); readln (k);
 min := A[k,1];
 for c:=2 to n do
  if A[k,c] < min then
  min:= A[k,c];
write (‘min. de pe linia’, k, ‘=’, min);
end;
	int k, min;
	{
	cout<<” specificati linia=” ; cin>>k;
	min=A[k][0];
	  for (c=1; c< n; c++)
	  if A[k][c] < min
	  min= A[k][c];
	cout<<”min. de pe linia”<<k<<”=”<<min;
	}

PASCAL	C / C++
D. Simularea unor situații reale. **Exemplu:** memorarea relațiilor de prietenie dintre membrii unui grup de n persoane (n<=10)	

var R : array [1..10, 1..10] of byte;
l,c,i,j,d,n : byte;
begin
write (‘ numarul de persoane=’);
readln (n);
d:=1;
repeat
write (‘prietenie intre: ’); readln (i,j);
{relatia de pritenie este reciproca}
R[i,j] :=1; R[j,i] :=1;
write (‘mai sunt prieteni? ’);
readln (d);
{ da (d=1)/ nu (d=0)}
until d=0;
for l:=1 to n do
begin
writeln;
for c:=1 to n do
write (R[l,c], ‘ ’);
end;
end;
	int R[10] [10] ;
	int l,c,i,j,d=1,n ;
	{
	cout<<” numarul de persoane=” ; cin>>n;
	while (d)
	{
	cout<<”prietenie intre:” ; cin>>i>>j;
	// relatia de prietenie este reciproca
	R[i][j]=1; R[j][i]=1;
	cout<<” mai sunt prieteni?” ; cin>>d;
	// da (d=1)/ nu (d=0)
	}
	for (l=0; l< n; l++)
	{
	cout<<endl;
	for (c=0; c< n; c++)
	cout>>R[l][c]<<” ”;
	}
	}

Reluăm aplicația Floare de colț

Din analiza problemei, a reieșit necesitatea organizării datelor într-un tablou bidimensional. Continuăm rezolvarea problemei.

2. *Raționamentul problemei*

Pentru determinarea temperaturii minime și a temperaturii maxime înregistrate în cele zece zone, pe parcursul săptămânii, se prelucrează toate elementele din tablou.

Pentru determinarea temperaturii medii, **tmed**, se solicită zona și se prelucrează doar elementele de pe linia corespunzătoare.

Pentru determinarea zonei în care s-a atins temperatura minimă sau maximă, se solicită ziua și se prelucrează doar coloana corespunzătoare.

3. *Reprezentarea algoritmului*

```
begin temperaturi2
alocă T[10, 7]
pentru zona = 1, 10 execută
    pentru zi = 1, 7 execută
        citește T[zona, zi]
    sfârșit pentru
sfârșit pentru
tmin ← T[1, 1]
tmax ← T[1, 1]
pentru zona = 1, 10 execută
    pentru zi = 1, 7 execută
        dacă T[zona, zi] < tmin
            atunci tmin ← T[zona, zi]
        sfârșit dacă
        dacă T[zona, zi] > tmax
            atunci tmax ← T[zona, zi]
        sfârșit dacă
    sfârșit pentru
sfârșit pentru
scrie tmin
scrie tmax
scrie "ce zonă vă interesează?"
citește zona
tmed ← 0
pentru zi = 1, 7 execută
    tmed ← tmed + T[zona, zi]
sfârșit pentru
tmed ← tmed/7
scrie zona, tmed
scrie "ce zi vă interesează?"
citește zi
tmin ← T[1, zi]
zmin ← 1
tmax ← T[1, zi]
zmax ← 1
pentru zona = 2, 10 execută
    bloc
        dacă T[zona, zi] < tmin
            atunci
                bloc
                    tmin ← T[zona, zi]
                    zmin ← zona
                sfârșit bloc
        sfârșit dacă
        dacă T[zona, zi] > tmax
            atunci
                bloc
                    tmax ← T[zona, zi]
                    zmax ← zona
                sfârșit bloc
        sfârșit dacă
    sfârșit bloc
sfârșit pentru
scrie zmin, tmin, zi
scrie zmax, tmax, zi
sfârșit temperaturi2
```

TEME

1. La cabinetul medical, se calculează înălțimea medie a tuturor băieților din școală. Precizați care este forma de organizare a datelor corespunzătoare acestei situații.

2. La cabinetul medical, se calculează înălțimea medie a tuturor băieților din școală, pe grupe de vârstă (ani de studiu). Precizați care este forma de organizare a datelor corespunzătoare acestei situații.

3. La cabinetul medical, se calculează înălțimea medie a tuturor băieților din școală pe grupe de vârstă (ani de studiu), pentru fiecare clasă în parte (9A,..., 12 C,...). Precizați care este forma de organizare a datelor corespunzătoare acestei situații.

4. Administratorul unui bloc cu 12 etaje și 20 de apartamente pe etaj calculează cheltuielile de întreținere, în funcție de numărul de persoane din fiecare apartament (toate apartamentele au același număr de camere).

Realizați un program care să răspundă următoarelor cerințe:

a) înregistrarea numărului de persoane din fiecare apartament, pe etaje, de la parter până la ultimul etaj;

b) afișarea numărului de persoane din fiecare apartament, pe etaje, de la ultimul etaj până la parter;

c) cunoscând numărul unui apartament, introdus de la tastatură, să se afișeze etajul la care se află apartamentul și numărul de persoane care locuiesc în apartamentul respectiv.

5. Se consideră tabloul **M** cu următoarele elemente:

1	2	3	4
5	6	7	8
9	10	11	12

Precizați ce valori afișează secvența de mai jos:

```
pentru c = 1, 3 execută
    pentru l = 1, 2 execută
        scrie M[l, c]
    sfârșit pentru
sfârșit pentru
```

a) 1, 5, 9, 2, 6, 10, 3, 7, 11; b) 1, 2, 3, 5, 6, 7; c) 1, 5, 2, 6, 3, 7.

6. Ce realizează următoarea secvență de operații:

```
alocă M[10, 10]
pentru i = 1, 10 execută
    M[i, i] ← i
sfârșit pentru
```

a) atribuie valori de la 1 la 10 elementelor din vectorul **M**;

b) atribuie fiecărui element de pe diagonala matricei **M** o valoare egală cu linia pe care acesta se află;

c) atribuie valori de la 1 la 10 primelor 10 elemente din matricea **M**.

7. Care dintre următoarele secvențe de operații memorează în colțurile matricei **M** valori citite de la tastatură; matricea are 5 linii și 5 coloane.

a)
```
pentru i = 1, N execută
    pentru j = 1, N execută
        citește M[i, j]
    sfârșit pentru
sfârșit pentru
```

b)
```
pentru i = 1, 5 execută
    pentru j = 1, 5 execută
        dacă (i = 1) și (j = 5)
            atunci citește M[i, i]
            altfel citește M[j, j]
        sfârșit dacă
    sfârșit pentru
sfârșit pentru
```

c)
```
l ← 1
c ← 5
pentru x = 1, 4 execută
    citește M[l, c]
    c ← l
    l ← c
sfârșit pentru
```

d)
```
l ← 1
c ← 5
citește M[l, l], M[l, c], M[c, l], M[c, c]
```

4. TABLOURI BIDIMENSIONALE – CAZURI PARTICULARE

Matrice pătrată

Rezolvarea problemelor cu calculatorul necesită găsirea soluțiilor de organizare și memorare a datelor, astfel încât acestea să păstreze semnificațiile reale atât din punct de vedere al valorilor proprii, cât și al relațiilor cu alte date. Spre exemplu, dacă membrii unui grup (persoane) împrumută bani unii de la alții, interesează atât suma de bani primită/datorată, cât și cine/de la cine a împrumutat. În acest caz, dacă în grup sunt *n* persoane, un tablou bidimensional, *G*, cu *n* linii și *n* coloane, este suficient pentru păstrarea atât a valorilor împrumutate, cât și a relațiilor de împrumut (fig. 8).

G	1	2	3	4	5
1			50		10
2	30				
3					
4		60			10
5			5		

ds

dp

Figura 8

Fiecare element din tablou reprezintă valoarea unui împrumut; liniile și coloanele reprezintă persoanele din grup care dau bani unei alte persoane sau primesc bani de la altă persoană din grup. Fie *i* și *j* două persoane: *G[i,j]* reprezintă suma de bani pe care i-a împrumutat-o lui *j,* adică suma de bani pe care *j* a primit-o de la *i*. Pentru exemplul din figura 8, *G[4,2]* reprezintă suma de 60 lei pe care persoana 4 a împrumutat-o persoanei 2.

Tabloul folosit are o particularitate: numărul de linii este egal cu numărul de coloane, de aceea este numit tablou pătrat sau, mai simplu, *matrice pătrată.*

Într-o matrice pătrată deosebim următoarele elemente specifice (fig. 8):

– diagonala principală (*dp*);
– diagonala secundară (*ds*);
– triunghiurile formate de cele două diagonale;
– direcțiile paralele cu fiecare dintre cele două diagonale.

Matrice binară

Reluăm situația grupului de persoane care împrumută bani unii de la alții, dar urmărim numai relația de împrumut: cine/de la cine a primit bani. În acest caz, nu se mai păstrează valoarea împrumutului. Fie *i* și *j* două persoane: *G[i,j]* are semnificația *i* a împrumutat bani lui *j* echivalent cu *j* a primit bani de la *i* (fig. 9). Elementele matricei nu pot avea decât două valori alese convențional (spre exemplu 0 sau 1), cu semnificația:

G	1	2	3	4	5
1	0	0	1	0	1
2	1	0	0	0	0
3	0	0	0	0	0
4	0	1	0	0	1
5	0	0	1	0	0

Figura 9

$$G[i,j] = \begin{cases} 1 & \text{dacă } i \text{ împrumută bani lui } j \\ 0 & \text{dacă } i \text{ nu împrumută bani lui } j \end{cases}$$

Matricea ale cărei elemente au valori în mulțimea {0,1}, cu semnificații logice complementare, se numește *matrice binară.*

Matrice simetrică

Dacă în grupul de *n* persoane urmărim relațiile de prietenie, acestea pot fi memorate tot într-o matrice binară ale cărei elemente au următoarea semnificație:

$$P[i,j] = \begin{cases} 1 \text{ dacă } i \text{ este prieten cu } j \\ 0 \text{ dacă } i \text{ nu este prieten cu } j \end{cases}$$

Prietenia este o relație reciprocă; acest aspect se regăsește în proprietatea de *simetrie* a matricei binare asociată grupului de persoane: *P [i, j] = P [j, i]* (fig. 10).

P	1	2	3	4	5
1			1	1	
2			1		1
3	1	1		1	
4	1		1		1
5		1		1	

Figura 10

Matricea punctelor unui plan

Pe ecranul monitorului, în modul de lucru text, caracterele sunt afișate pe rânduri, de sus în jos, de la stânga la dreapta, pe fiecare rând. Se poate spune că ecranul monitorului este o suprafață plană ale cărei puncte sunt distribuite pe linii și coloane (cel mai frecvent, 24 de linii și 80 de coloane). În acest exemplu, regăsim modelul bidimensional de organizare a datelor: oricărei suprafețe plane îi poate fi asociată o *matrice a punctelor*. Valorile atribuite elementelor din matrice au semnificația specifică problemei modelate. Pentru exemplul suprafeței-ecran, dacă elementele matricei sunt de tip caracter, în matrice poate fi reținut textul de pe un ecran.

Un alt exemplu: o imagine – fotografie – este tot o reprezentare în plan. Punctele planului formează obiecte distincte, dacă sunt evidențiate diferit de la un obiect la altul, prin culoare. Dacă toate punctele unei imagini (suprafață) sunt colorate la fel, imaginea este formată dintr-un singur obiect.

Oricărei imagini îi poate fi asociată o matrice a punctelor; valorile atribuite elementelor din matrice au semnificația culorii fiecărui punct. În figura 11 este reprezentată matricea asociată unei imagini, în care s-au folosit 3 coduri de culori cu semnificația: 0 – alb, 1– negru, 2 – roșu.

Cu ajutorul matricelor de tip plan, pot fi modelate și situații de joc: așezarea pieselor pe tabla de șah, configurația unui labirint, „X și O”, „avioane” și altele.

0	0	0	1	0	0	0	0
0	1	1	1	1	1	0	0
1	2	2	2	2	0	0	0
1	2	2	2	1	0	0	0
1	2	2	1	2	0	0	0
0	1	1	2	1	1	1	0
0	0	1	2	1	2	1	0
0	1	1	1	0	1	1	0

Figura 11

TEME

1. Determinați proprietățile elementelor aflate pe diagonala principală, într-o matrice pătrată. Scrieți un program pentru afișarea acestor elemente.

2. Determinați proprietățile elementelor aflate pe diagonala secundară, într-o matrice pătrată. Scrieți un program pentru afișarea acestor elemente.

3. Scrieți un program care să verifice dacă o matrice pătrată este simetrică.

4. Formulați un exemplu de problemă care să necesite organizarea datelor într-o matrice binară.

5. Formulați un exemplu de problemă care să necesite organizarea datelor într-o matrice de tip plan.

6. Determinați condițiile de amplasare pe tabla de șah a două piese de joc – dame –, astfel încât acestea să nu se atace. (Indicație: două piese de șah – dame – se atacă dacă sunt amplasate pe aceeași linie, pe aceeași coloană sau pe aceeași diagonală.)

7. Construiți tabloul de vecinătate pentru țările situate pe harta din figura 12.

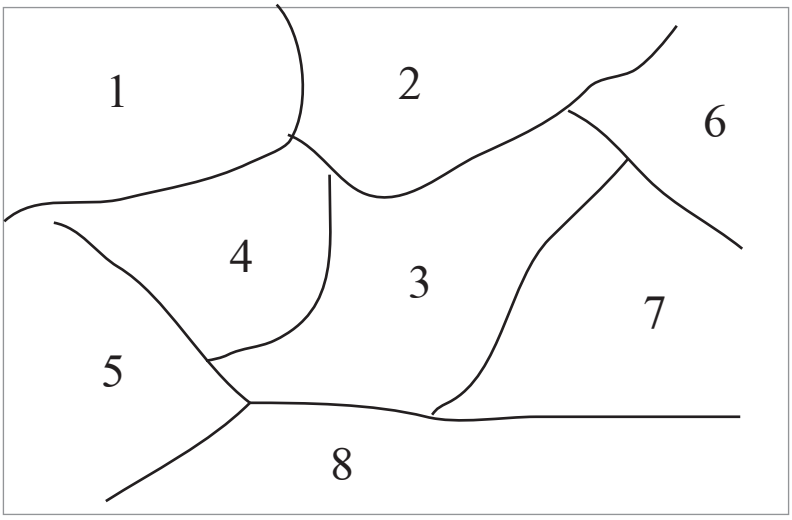


Figura 12

8. Să se verifice dacă o matrice pătrată este „tablou magic”. Într-un tablou magic, suma elementelor de pe oricare linie este egală cu suma elementelor de pe oricare dintre coloane precum și cu suma elementelor de pe oricare dintre diagonale.

Exemplu:

3	2	7
8	4	0
1	6	5

5. PRELUCRAREA TABLOURILOR BIDIMENSIONALE

5.1. Localizarea elementelor cu aceeași proprietate

Formațiuni geografice

Se dorește determinarea configurației unui teren dreptunghiular după formațiunile geografice din Tabelul 3 și figura 13.

Analiza terenului se face prin secționarea acestuia pe orizontală și pe verticală.

Punctele aflate la intersecția dintre secțiunile orizontale și secțiunile verticale sunt cotate față de nivelul mării; cotele sunt valori numerice întregi și pozitive.

*Terenul este secționat prin **n** secțiuni orizontale și **m** secțiuni verticale.*

O formațiune geografică este formată din cel puțin trei puncte.

Tabelul 3. **Formațiuni geografice**

FORMAȚIUNEA GEOGRAFICĂ	
Pantă	a)
Râpă	b)
Deal	c)
Vale	d)
Platou	e)
Teren denivelat	f)
Punct de tip *șa_xy*: punct aflat la cota maximă pe secțiunea orizontală *x* și la cota minimă pe secțiunea verticală *y*; se poate defini și punct de tip *șa_yx*	g)

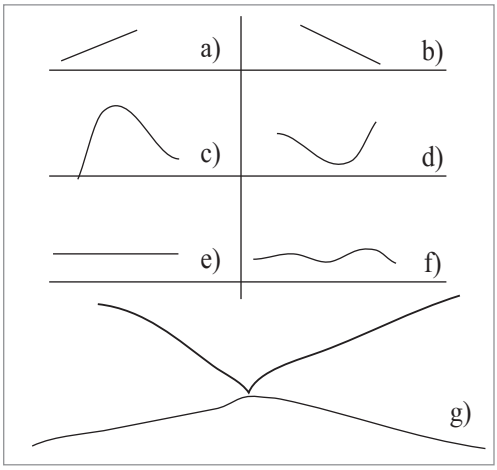


Figura 13

Exemplu:

În Tabelul 4 este reprezentat un teren pe care s-au făcut patru secțiuni orizontale și șapte secțiuni verticale.

Tabelul 4. **Înregistrarea datelor într-un teren secționat**

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	25	72	69	69	69	73	40
2	20	40	50	56	30	19	100
3	15	30	35	40	39	20	10
4	10	55	0	60	42	50	19

Rezultatele analizei pe secțiuni sunt prezentate în tabelul Tabelul 5.

Tabelul 5. **Analiza pe secțiuni**

Secțiuni orizontale		Secțiuni verticale	
Numărul secțiunii	**Formațiunea geografică**	**Numărul secțiunii**	**Formațiunea geografică**
1	Platou la cota 69	1	Râpă
2	Deal cu vârf la cota 56	2	Vale cu punct minim la cota 30
3	Deal cu vârf la cota 40	3	Râpă
4	Teren denivelat	4	Vale cu punct minim la cota 40
		5	Teren denivelat
		6	Vale cu punct minim la cota 19
		7	Teren denivelat
Puncte de tip șa_xy: punctul de coordonate [3,4] la cota 40			

Sugestie de rezolvare

Pentru determinarea formațiunilor geografice, cotele vor fi înregistrate într-un tablou bidimensional cu semnificația: linii – secțiuni orizontale, coloane – secțiuni verticale.

Pentru fiecare secțiune (linie sau coloană) se va studia monotonia șirurilor de valori (cotele înregistrate pe o linie sau pe o coloană) – Tabelul 6.

Rezolvarea problemei conduce la determinarea elementelor cu aceeași proprietate dintr-un tablou bidimensional.

Tabelul 6. **Monotonia șirurilor de valori**

Formațiunea geografică	Monotonia șirului de valori
Pantă/ Râpă	Șir crescător/ descrescător
Deal	Există un punct de tip vârf, astfel încât toate punctele dispuse la stânga acestuia formează un șir monoton crescător, iar toate punctele dispuse la dreapta acestuia formează un șir monoton descrescător.
Vale	Există un punct de cota minimă, astfel încât toate punctele dispuse la stânga acestuia formează un șir monoton descrescător, iar toate punctele dispuse la dreapta acestuia formează un șir monoton crescător.
Platou	Există cel puțin trei puncte consecutive aflate la aceeași cotă.

Pentru exemplificarea rezolvării, prezentăm, în pseudocod, secvența prelucrărilor pentru determinarea formațiunii de tip *platou* și a punctelor de tip *șa_xy* .

```
Determinarea formațiunii de tip platou
begin platou
// căutarea unei formațiuni platou pe linia k
//semnificația variabilelor de lucru
//M[100,100]  tabloul cotelor cu  m secțiuni verticale
// lp  lungimea platoului
// cp  cota platoului
// inițalizare lungime platou
lp←1
cp←M[k,1]
pentru c=2 la m execută
// se verifică dacă punctul M[k,c] aparține platoului
    dacă  M[k,c] =cp
            atunci lp ← lp+1   // crește  lungimea platoului
            altfel
                dacă  lp>=3     // există platou la copta cp
                    atunci  scrie platou la cota cp
                sfârșit dacă
// inițializări pentru determinarea unui nou platou
          lp←1
          cp←M [k,c]
    sfârșit dacă
sfârșit pentru
// se verifică dacă linia  k se termină cu platou
dacă lp>=3
   atunci  scrie platou la cota cp
sfârșit dacă
sfârșit platou
```

Determinarea punctelor de tip șa_xy

- Raționamentul de rezolvare

Pasul 1

Se parcurge tabloul pe linii: pentru fiecare linie, se determină elementul maxim și se păstrează coloana acestuia în vectorul *max_linii*.

Pentru exemplul dat, vectorul *max_linii* are următoarele valori: 6, 7, 4, 4.

Pasul 2

Se parcurge tabloul pe coloane; pentru fiecare coloană, se determină elementul minim și se păstrează linia acestuia în vectorul *min_coloane*.

Pentru exemplul dat, vectorul *min_coloane* are următoarele valori: 4, 3, 4, 3, 2, 2, 3.

Pasul 3

Se parcurge vectorul *max_linii*: în variabila de lucru *cmax,* se reține valoarea unui element *max_linii[k]*

cmax = max_linii[k]

se verifică proprietatea de punct șa_xy:

min_colane[cmax] = k

Pentru exemplul dat, urmărim datele din Tabelul 7.

Există punct șa_xy pe linia 3, coloana 4.

Tabelul 7. **Determinarea punctului șa_xy**

linia	Cmax	min-coloane[cmax]
1	6	2
2	7	3
3	4	3
4	4	3

- Secvența pseudocod a prelucrărilor pentru determinarea punctelor de tip *șa_xy*:

```
begin   punct_șa_xy
// căutarea unui punct șa_xy
//semnificația variabilelor de lucru
//M[100,100]  tabloul cotelor cu  m secțiuni orizontale  și n secțiuni verticale
// max_linii [100]  vector  cu m elemente în care se reține coloana  elementului maxim de pe
        fiecare linie
// min_coloane[100]  vector  cu n elemente în care se reține linia  elementului minim de pe
        fiecare coloană
//max    valoarea maximă   pe o linie;  cm coloana pe care se află  max
//min    valoarea minimă   pe o coloană;  lm linia  pe care se află  min

// se determină elementul maxim de pe fiecare linie
pentru k=1 la m execută
  max ←  M[k,1]
  cm  ← 1
  pentru c=2  la  n execută
   dacă  M[k,c] >max
            atunci
                 bloc
                 max← M[k,c]
                 cm ← c
                sfârșit bloc
    sfârșit dacă
  sfârșit pentru
```

```
// în linia k, elementul maxim se află pe coloana cm
max_linii[k] ←   cm
sfârșit  pentru

// se determină elementul minim de pe fiecare coloană
pentru c=1 la n execută
   min ←  M[1,c]
   lm  ← 1
   pentru k=2 la m execută
    dacă  M[k,c] <min
              atunci
                    bloc
                    min ← M[k,c]
                    lm ← k
                   sfârșit bloc
     sfârșit dacă
  sfârșit pentru
// în coloana  c,  elementul minim se află pe linia  lm
min_coloane[c] ←  lm
sfârșit pentru
// se parcurge vectorul max_linii
pentru k=1 la m execută
 cmax ← max_linii [k]
// se verifică proprietatea de punct șa_xy
 dacă  min_coloane[cmax]=k
              atunci
                          scrie  punct sa de coordonate k, cmax
  sfârșit dacă
sfârșit pentru
sfârșit punct _sa_xy
```

TEME

1. Construiți exemple numerice pentru următoarele formațiuni geografice:

a) platou pe coloana 3, c) râpă pentru coloana 5, e) punct șa_xy,
b) deal pentru linia 1, d) vale pentru linia 4, f) punct șa_yx.

2. Construiți expresii pentru relațiile de monotonie corespunzătoare fiecărei formațiuni geografice.

3. Codificați, în limbajul de programare studiat, secvența pseudocod pentru determinarea formațiunii geografice de tip platou.

4. Codificați, în limbajul de programare studiat, secvența pseudocod pentru determinarea punctelor de tip șa_xy.

5. Realizați, în limbajul de programare studiat, un program pentru determinarea punctelor de tip șa_yx.

6. Scrieți secvențele de instrucțiuni pentru determinarea următoarelor formațiuni geografice:

a) râpă, b) deal, c) vale.

7. Realizați și testați programul pentru derminarea configurației unui teren ale cărui coordonate și cote se citesc din fișierul *teren.in* cu următoarea structură:

– pe prima linie valorile *n* și *m* reprezentând: *n* numărul de secțiuni orizontale și *m* numărul de secțiuni verticale;

– pe următoarele *n* linii câte *m* valori reprezentând cotele aflate pe fiecare secțiune orizontală.

5.2. Prelucrarea elementelor distribuite pe aceleași direcții (linii, coloane, diagonale)

Aranjamente florale

Un grădinar are mai multe soiuri de plante pe care dorește să le planteze atât în aranjamente clasice, cât și în forme noi; spre exemplu, în locul rondurilor, grădinarul vrea să compună careuri florale. Întrucât timpul de creștere și înflorire al plantelor nu poate fi întârziat, grădinarul și-a propus să testeze modelele cu calculatorul.

Analiza problemei

Pentru rezolvarea cu calculatorul, soiurile de plante decorative vor fi codificate. În Tabelul 8 se prezintă un exemplu de codificare.

***Tabelul 8.* Codificarea plantelor decorative**

Planta decorativă/culoare	Codul plantei
Iarbă/verde	1
Trandafir imperial/roșu	2
Narcise/albe	6
Narcise/galbene	7
Crizanteme/mov	8
Crizanteme/albe	9
Tuia/verde	11

Careul pe care va fi probat modelul este format din *n***m* sau *n***n* puncte; în fiecare punct, poate fi sădită o plantă. În Tabelul 9 sunt prezentate câteva modele după care se vor testa aranjamentele florale.

Tabelul 9. **Modele pentru aranjamente florale**

Denumirea modelului	Exemplu de model
Brazde paralele orizontale	Decor de primăvară cu narcise albe și galbene: 6 6 6 6 6 7 7 7 7 7 6 6 6 6 6
Triunghi	Decor de primăvară cu narcise albe și galbene; aleea centrală (diagonală) cu gazon: 1 7 7 7 7 6 1 7 7 7 6 6 1 7 7 6 6 6 1 7 6 6 6 6 1
Diagonale paralele cu diagonala principală	Decor cu trandafiri imperiali și tuia plantați pe direcții paralele cu aleea centrală (diagonală): 1 2 11 2 11 2 1 2 11 2 11 2 1 2 11 2 11 2 1 2 11 2 11 2 1
Spirală	Decor de toamnă cu crizanteme mov și albe: 8 8 8 8 8 8 8 9 9 9 9 8 8 9 8 8 9 8 8 9 8 8 9 8 8 9 9 9 9 8 8 8 8 8 8 8

Sugestie de rezolvare

Careurile florale vor fi generate într-un tablou bidimensional, G, cu *n* linii și *m* coloane, sau *n* linii și *n* coloane, în funcție de model. Pentru fiecare model, se determină adresele punctelor în care vor fi sădite plantele și se înregistrează la aceste adrese codul plantei corespunzător modelului.

Rezolvarea problemei conduce la umplerea matricei cu valori dispuse pe direcții specifice modelului: linii, coloane, triunghiuri, diagonale, spirală.

În Tabelul 10 sunt prezentate secvențele pseudocod pentru generarea adreselor de umplere corespunzătoare modelelor prezentate.

Tabelul 10. **Secvențe pseudocod pentru generarea adreselor de umplere**

Modelul floral	Direcțiile de umplere	Generarea adreselor

Brazde orizontale	Linii	// brazda de tip linie
		// linia i
		pentru c=1 la m **execută**
		G[i,c] ← cod
		sfârșit pentru

Triunghi	Triunghi stânga	//se lucrează cu o matrice pătrată **n*n**
		// pe linii **i** și coloane **j**
		pentru i=2 la n **execută**
		pentru j=1 la i-1 **execută**
		G[i,j] ← cod
		sfârșit pentru
		sfârșit pentru

Paralele la diagonala principală	Se pot forma n-2 direcții **(d)** paralele cu diagonala principală și situate deasupra acesteia	//se lucrează cu o matrice **n*n**
		// pe linii **i** și coloane **j**
		// pe diagonalele **d**
		
		pentru d=1 la n-2 **execută**
		pentru i=1 la n-d **execută**
		G[i,i+d] ← cod
		sfârșit pentru
		sfârșit pentru

Spirală	Se parcurg cadranele de la exterior – primul cadran-**p**, spre interior, ultimul cadran-**u**	u p←1 u←n
		repetă
		// se parcurge prima linie, **p**, din cadran,
		//de la stânga la dreapta
		pentru j=p **la** u **execută**
		G[p,j] ← cod
		sfârșit pentru
		// se parcurge ultima coloană, **u**, din cadran,
		// de sus în jos
		pentru i=p+1 **la** u **execută**
		G[i,u] ← cod
		sfârșit pentru
		//se parcurge ultima linie, **u**, din cadran

		//de la dreapta la stânga
		pentru j=u -1 **la** p **execută**
		G[u,j] ← cod
		sfârșit pentru
		// se parcurge prima coloană, **p**, din cadran,
		// de jos în sus
		pentru i=u - 1 **la** p - 1 **execută**
		G[i,p] ← cod
		sfârșit pentru
		
		// se pregătesc valorile **p** și **u**
		// pentru cadranul următor
		p← p+1
		u←u-1
		// se testează dacă se mai pot forma cadrane
		până când p>u

TEME

1. Construiți expresii pentru relațiile care definesc direcțiile de umplere pentru fiecare dintre modelele propuse.

2. Scrieți secvențele de instrucțiuni pentru generarea fiecărui model.

3. Realizați și testați programul pentru generarea fiecărui model floral (pentru fiecare model se va scrie un program).

4. Pentru atractivitatea prezentării, realizați un program care să afișeze modelul floral colorat, corespunzător culorii de cod. (Indicație: modul de lucru text permite setarea atributului de culoare atât pentru fond, cât și pentru text.)

5. Realizați un program care să permită utilizatorului (grădinarul) să aleagă, pe rând, oricare dintre modelele oferite (program cu meniu – fig. 14).

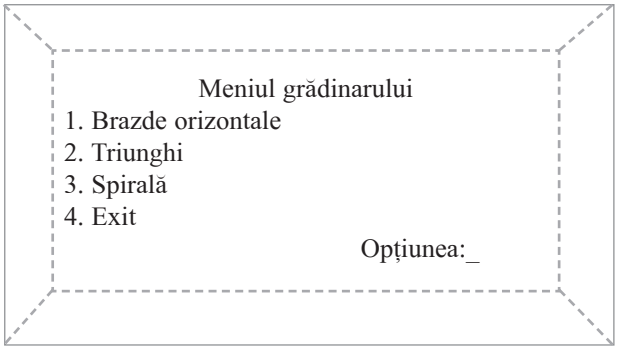


Figura 14

6. Rescrieți secvența umplerii în spirală astfel încât să folosiți cât mai puține structuri repetitive.

7. Compuneți un model floral nou și scrieți secvența de instrucțuni (programul) pentru generarea modelului propus.

8. Analizați următoarele secvențe pseudocod și determinați modelul de umplere generat:

a)
pentru i=2 **la** n **execută**
  pentru j=n, **la** n+2-i **execută**
  G[i,j] ← cod
  sfârșit pentru
sfârșit pentru
	b)
	pentru i= 1 **la** [n/2] **execută**
	  pentru j=i+1 **la** n-i **execută**
	  G[i,j] ← cod
	  sfârșit pentru
	sfârșit pentru

c)
pentru i=n **la** [n/2]+1 **execută**
  pentru j=n **la** n+2-i **execută**
  G[i,j] ← cod
  sfârșit pentru
sfârșit pentru
	d)
	pentru i=1 **la** n **execută**
	  pentru j=1, la **m execută**
	  G[i,j] ← cod
	  sfârșit pentru
	sfârșit pentru

5.3. Simularea unor situații reale

Tabloul de familie

Se consideră o familie formată din părinți (1) și copii (2). Fiecare copil are doi părinți între care există relație de căsătorie. Nu toți membrii familiei sunt căsătoriți; nu toți membrii familiei au copii.

Relațiile dintre membrii familiei sunt păstrate în tabloul de familie (fig. 15).

Figura 15. Tablou pentru o familie formată din 8 persoane

Membrul familiei	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***
1				*1*		*2*		*2*
2					*2*	*1*	*2*	
3					*1*			
4	*1*					*2*		*2*
5			*1*					
6		***1***			***2***		***2***	
7								
8								

Interpretarea tabloului de familie este prezentată în Tabelul 11.

Tabelul 11.

Membrul familiei	Căsătorit cu	Copii	Părinți
1	*4*	*6 și 8*	
2	*6*	*5 și 7*	
3	*5*		
4	*1*	*6 și 8*	
5	*3*		*2 și 6*
6	***2***	***5 și 7***	***1 și 4***
7			***2 și 6***
8			***1 și 4***

După cum rezultă din Tabelul 11, fiecare membru al familiei este complet caracterizat prin analiza înregistrărilor de tip linie și a înregistrărilor de tip coloană.

Analiza înregistrărilor de tip linie:

• familia are 8 membri; pentru fiecare membru (*i*) analizăm valorile de pe linia *i* din tablou, cu semnificația: (1) – *i* este căsătorit cu *j,* unde *j* este poziția valorii 1 pe linie,
(2) – *i* este părinte pentru *k,* unde *k* este poziția valorii 2 pe linie.

Analiza înregistrărilor de tip coloană:

• familia are 8 membri; pentru fiecare membru (*j*) analizăm valorile de pe coloana *j* din tablou cu semnificația: (1) – *j* este căsătorit cu *i,* unde *i* este poziția valorii 1 pe coloană,
(2) – *k* este părinte pentru *j,* unde *k* este poziția valorii 2 pe coloană.

Concluzii și restricții

- oricărei familii îi poate fi asociat un tablou de familie de tip matrice pătrată;
- pe o linie poate fi înregistrată cel mult o valoare 1;
- pe o coloană poate fi înregistrată cel mult o valoare 1;
- pe o coloană pot fi înregistrate cel mult două valori 2;
- dacă *i* este în relație de tip 1 cu *j,* și *j* este în relație de tip 1 cu *i;*
- dacă *i* este în relație de tip 1 cu *j,* și *i* este în relație de tip 2 cu *k,* atunci și *j* este în relație de tip 2 cu *k.*

TEME

1. Cunoscând relațiile de tip 1 sau 2 dintre cei *n* membri ai unei familii, să se construiască tabloul de familie.

2. Validarea tabloului de familie: fiind dat un tablou de familie, să se verifice corectitudinea înregistrărilor.

3. Fiind dat un tablou de familie validat, caracterizați fiecare membru al familiei.

4. Să se determine descendenții unui membru al familiei (pentru exemplul analizat, 1 are copii pe 6 și 8 și nepoți pe 5 și 7).

5. a) Realizați tabloul de familie pentru familia cu 10 membri din exemplul următor:

Membrul familiei	Căsătorit cu	Copii
2	3	1, 5, 7
5	8	1, 10
10	4	9

b) Analizați tabloul de familie obținut la cerința **a)** și precizați ce restricții au fost încălcate.

6. Cum procedăm pentru a determina toți descendenții unui membru al familiei?

7. Cum procedăm pentru a determina cel mai vârstnic ascendent al familiei?

5.4. Prelucrarea imaginilor

Aplicația 3. Virusuri într-o matrice

După ce au descoperit cele mai puternice virusuri, biologii și-au propus să le izoleze și să le studieze comportamentul în alte colonii de bacterii. După o vreme, au constatat că unele virusuri au rămas izolate, altele nu. Privită la microscop, colonia de bacterii pare formată dintr-o mulțime de puncte pe care biologii le-au reprezentat ca în figura de mai jos.

În această colonie există un singur virus izolat (cel încercuit). Biologii doresc să prelucreze aceste „imagini” cu calculatorul. Pentru început vor să localizeze și să numere virusurile izolate.

Cum rezolvăm problema?

1. Introducerea imaginii

De data aceasta, datele de intrare au aspectul unei „imagini alb-negru”: punctele de interes sunt virusurile colorate în negru; restul suprafeței este colorată în alb. Pentru a transmite calculatorului „imaginea”, ar trebui să-i spunem culoarea fiecărui punct. Sau, mai simplu, să-i spunem de la început că imaginea este „albă” și apoi să-i dăm adresele punctelor colorate în „negru”. Imaginea descompusă în puncte seamănă atât de bine cu o matrice, încât ea poate fi memorată codificând albul cu zero și negrul cu unu.

	x				x
x			x	x	x
			x	x	
	(x)		x		

0	1	0	0	0	1
1	0	0	1	1	1
0	0	0	1	1	0
0	1	0	1	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Localizarea virusurilor izolate

2. Reprezentarea algoritmului

Secvența de operații pentru citirea imaginii:

```
început citire
    alocă M[10,10]
    citește NL, NC      //număr linii -NL
                        //număr coloane -NC
    pentru i = 1, NL execută
        pentru j = 1, NC execută
            M[i, j] ← 0
        sfârșit pentru
    sfârșit pentru
    citește p //numărul de puncte negre
    pentru k = 1,p execută
    bloc
        citește l,c
        M[l,c] ← 1
    sfârșit bloc
    sfârșit pentru
sfârșit citire
```

3. Prelucrarea imaginii. Bordajul

Urmează prelucrarea „imaginii”: localizarea și numărarea virusurilor (punctele negre izolate).

Pentru rezolvarea acestei cerințe, se parcurge matricea și, pentru fiecare valoare de 1, se verifică vecinii acesteia. Un punct „negru” este izolat dacă toți vecinii săi sunt „albi”.

Câți vecini are un punct? Majoritatea punctelor au câte opt vecini. Punctele de la „bariera” coloniei au mai puțini vecini.

Pentru a simplifica procedeul de numărare a vecinilor, vom înconjura colonia cu o zonă fără virusuri. Imaginea se lărgește. La fel și matricea: în urma acestui „bordaj”, ea se va mări cu două linii și două coloane.

0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	1	1	1	0
0	0	0	0	1	1	0	0
0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Bordajul se pregătește înainte de citirea imaginii prin alocarea unui spațiu de memorie suficient, ținând seama că la numărul de linii și coloane necesar memorării imaginii se mai adaugă câte două pentru bordaj.

În matricea bordată, fiecare punct al imaginii are câte opt vecini. În figura 16 sunt prezentate adresele vecinilor punctului aflat la adresa [**l**, **c**].

l-1, c-1	l-1, c	l-1, c+1
l, c-1	l, c	l, c+1
l+1, c-1	l+1, c	l+1, c+1

Figura 16. Vecinii unui punct din matrice

Dacă punctul aflat la adresa [**l**, **c**] are valoarea 1 și fiecare dintre cei opt vecini ai săi are valoarea zero, atunci punctul este izolat. Condiția o vom nota cu **cond**.

4. Reprezentarea algoritmului

• Secvența pentru localizarea și numărarea punctelor izolate:

```
begin puncte_izolate
    nrp ← 0
    pentru l = 2,NL + 1 execută
        pentru c = 2,NC+1  execută
            dacă M[l,c] = 1 atunci
                dacă cond
                    atunci
                        bloc    scrie l - 1, c - 1
                                nrp ← nrp + 1
                        sfârșit bloc
                sfârșit dacă
            sfârșit dacă
        sfârșit pentru
    sfârșit pentru
    scrie nrp
sfârșit puncte_izolate
```

TEMĂ

Folosind cele două secvențe prezentate, realizați algoritmul complet și programul corespunzător.

De reținut!

1. Cu ajutorul matricelor pot fi rezolvate probleme care necesită descompunerea unei imagini în puncte sau probleme de orientare și deplasare în plan.

2. Pentru ușurința prelucrării punctelor se folosește tehnica bordajului.

3. Multe jocuri precum „X și zero”, „Vaporașe”, „Perspico”, mutarea pieselor pe tabla de șah, „Război” pot fi simulate cu ajutorul matricelor.

PROBLEME PROPUSE

1. JOC

Se dau *n* jetoane numerotate de la 1 la *n*. Din fiecare tip de jeton, există *n* bucăți. Jetoanele trebuie așezate pe o grilă formată din *n***n* careuri, ca în exemplul din figura 17 a.

Figura 17. *Joc cu jetoane*

a)

1	2	3	4	5	6
2	3	4	5	6	1
3	4	5	6	1	2
4	5	6	1	2	3
5	6	1	2	3	4
6	1	2	3	4	5

b)

1	2	3	4	5
2	3	4	5	1
3	4	4	1	2
4	5	1	2	3
5	1	2	3	4

Cerințe:

1. Stabiliți regula după care sunt așezate jetoanele pe grila de joc.
2. Realizați un program care să așeze jetoanele pe grila de joc după regula stabilită la cerința precedentă.
3. Realizați un program care să verifice dacă jetoanele de pe grila de joc respectă regula de amplasare stabilită la cerința 1.

Spre exemplu, pentru grila de joc din figura 17 b, nu sunt îndeplinite următoarele proprietăți:

- pe diagonala secundară se află un jeton diferi de jetonul *n*;
- pe linia 3 nu se află jetoane distincte;
- pe coloana 3 nu se află jetoane distincte.

Indicație: se vor analiza jetoanele dispuse pe aceeași linie/coloană sau pe diagonale/paralele la diagonale.

2. TABELE MATEMATICE

a) TABLA ÎNMULȚIRII

Realizați un program pentru afișarea sub formă de tabel a tablei înmulțirii cu 1, 2, 3, până la 10. Spre exemplu, linia 3 va conține tabla înmulțirii cu 3:

3	6	9	12	15	18	21	24	27	30

b) TABLA ADUNĂRII

Realizați un program pentru afișarea sub forma de tabel a tablei adunării cu 1, 2, 3, până la 10. Spre exemplu, linia 7 va conține tabla adunării cu 7:

8	9	10	11	12	13	14	15	16	17

c) TABELA PITAGORA_n

Realizați un program pentru afișarea tabelei *Pitagora_n* sub formă de tabel cu *n* linii și trei coloane: *a, b, c*. Pentru fiecare linie, valorile din tabelă trebuie să respecte condiția:

*a*a = b*b + c*c*. **Exemplu:** tabela *Pitagora_2*

5	3	4
15	12	9

d) MATRICEA *n_PALINDROM*

Realizați un program care să verifice dacă o matrice este *n_Palindrom*. O matrice *n_Palindrom* are *n* linii; pe fiecare linie, *i*, se află câte *n* palindromuri distincte formate din *i* cifre. Un număr simetric se numește *palindrom.* Exemplu de matrice *5_Palindrom:*

1	9	7	6	3
22	33	66	4455	55
121	323	464	585	979
3223	8668	7337	8118	9009
56365	43234	11111	91219	37473

3. MATRICE RARĂ

O matrice cu *n* linii și *m* coloane se numește *matrice rară* dacă valorile 0 sunt majoritare. **Exemplu** pentru o matrice cu 3 linii și 5 coloane:

0	0	20	0	0
0	0	0	12	0
0	88	0	0	0

Într-o astfel de matrice, interesează valorile semnificative, de aceea este suficient să memorăm valorile diferite de zero și pozițiile lor în matrice. Poziția reprezintă în acest caz numărul de ordine de 1 la *n*m*.

Pentru exemplul prezentat, se vor memora perechile: (20, 3) , (12, 9) și (88,12).

Cerințe:

a) propuneți structurile de date necesare memorării valorilor semnificative dintr-o matrice rară;

b) se cunoaște numărul valorilor semnificative dintr-o matrice rară (fie *p* acest număr) și perechile *valoare*, *adresă* pentru fiecare număr semnificativ.

Scrieți un program care să genereze și să afișeze pe ecran matricea rară.

Exemplu: Date de intrare

n =4 m=4 p=2

23 4 9 14

Date de ieșire — matricea rară

0	0	0	23
0	0	0	0
0	0	0	0
0	9	0	0

6. ORGANIZAREA DATELOR ÎN STRUCTURI NEOMOGENE

6.1. Studiu de caz *Catalogul clasei*

Într-o clasă sunt 30 de elevi. Profesorul diriginte dorește să păstreze, într-un catalog electronic, următoarele informații (figura 18):

– numele elevului;
– media generală pe semestrul I;
– media generală pe semestrul II;
– media generală anuală;
– numărul de absențe.

Profesorul diriginte dorește să calculeze media generală a clasei la sfârșit de semestru sau an școlar și să afișeze:

– media generală a clasei;
– lista elevilor după media generală;
– lista elevilor după numărul de absențe.

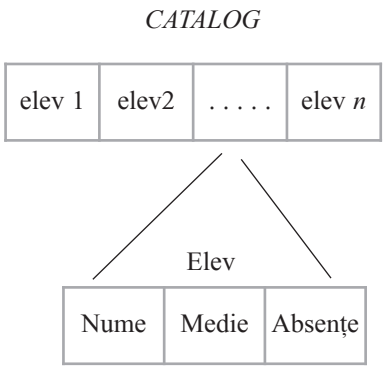


Figura 18

Analiza problemei

În catalogul electronic se vor păstra date despre elevi; se poate spune că în catalogul electronic se vor păstra date cu aceeași semnificație. Numărul înregistrărilor din catalog va fi egal cu numărul de elevi din clasă. Se va ține seama de numărul maxim de elevi ce pot fi înscriși într-o clasă (30 de elevi).

Catalogul poate fi organizat ca un tablou unidimensional – vector de elevi.

Pentru fiecare elev se rețin mai multe categorii de date, cu semnificații diferite – *date neomogene*. Fiecare informație are un tip specific, spre exemplu:

– medie de tip *real*;
– număr de absențe de tip *întreg* cu valori pozitive.

Soluția de organizare a datelor

Ansamblul datelor despre *elev* formează o structură de date cu tipuri diferite (structură neomogenă) numită *articol* sau *înregistrare*.

Catalogul electronic al clasei va fi un *vector de articole*.

Prelucrarea datelor

Pentru rezolvarea cerințelor formulate de profesorul diriginte, sunt necesare următoarele prelucrări:

– înregistrarea datelor pentru fiecare elev;
– afișarea datelor despre elevi;
– determinarea și afișarea mediei generale a clasei la sfârșit de semestru;

– afișarea elevilor în ordine descrescătoare, după media generală semestrială;
– determinarea mediei generale, anuale, pentru fiecare elev;
– afișarea elevilor în ordine descrescătoare, după media generală anuală;
– determinarea și afișarea mediei generale a clasei la sfârșit de an;
– afișarea elevilor în ordine crescătoare, după numărul de absențe.

6.2. Definirea structurilor neomogene de date-articole

Categoriile distincte de informații reținute într-un articol se numesc *câmpuri*.

Un articol se caracterizează prin următoarele elemente ce formează *macheta articolului*:

– numele articolului;

– două sau mai multe *câmpuri*; pentru fiecare *câmp* se precizează:

– numele câmpului;

– tipul câmpului (stabilit în funcție de semnificația reală a datelor). Câmpurile pot fi date elementare sau grupate: *vectori*, *articole*.

Exemplul 1:

Pentru exemplul analizat – catalogul clasei –, construim o machetă simplificată (cu mai puține câmpuri) corespunzătoare articolului *elev* (fig. 19a).

Elev	
Numele câmpului	Tipul câmpului
MEDIE	Real
ABSENȚE	Natural

Figura 19, a)

Exemplul 2:

Completăm macheta articolului *elev* cu câmpul note, un tablou unidimensional (*vector)* cu 4 elemente de tip natural (fig. 19b).

Elev	
Numele câmpului	Tipul câmpului
NOTE	Vector (4)
MEDIE	Real
ABSENȚE	Natural

Figura 19, b)

Fiecare limbaj de programare dispune de cuvinte cheie pentru declararea structurilor de date de tip articol (Tabelul 12).

Tabelul 12. **Definirea articolelor**

PASCAL	C/C++
*{cuvântul cheie pentru tipul articol este **record**}* *Type* **articol = record** camp1:*tip1*; camp2:*tip2*; ——————————— end;	*// cuvântul cheie pentru tipul articol // este **struct*** *Typedef* struct { *tip1* camp1; *tip2* camp2; ———————————- } **articol**;
{definirea variabilei cu tipul asociat articolului} *var* variabila : **articol**;	//definirea variabilei cu tipul asociat articolului **articol** variabila;
Exemplu pentru articolul elev	

{ se definește articolul elev }
Type elev=record
……
medie:*real*;
absente:*byte*;
. . . . . .
end;
 	
{se definește variabila cu tipul elev}
 	
var e: *elev*;
	// se definește articolul elev
	Typedef struct
	{
	……
	float medie;
	int:absente;
	. . . . . .
	} elev;
	// se definește variabila cu tipul elev
	
	elev e;

TEME

1. Realizați macheta articolului *data calendaristică* în care să fie reținută data în forma *zi, lună, an*.

2. Realizați macheta articolului *elev* care să conțină și câmpul *data nașterii* sub forma grupului de date *data calendaristică*.

3. Realizați macheta articolului *elev* care să conțină câmpul *medii* sub forma unui vector.

6.3. Prelucrarea datelor organizate în structuri neomogene

Prelucrarea datelor înregistrate într-o structură de tip articol se face la nivel de câmp. Pentru accesul la informațiile unui câmp, referirea acestuia se face prin *adresare punctuală* după sintaxa: *nume-variabilă-de-tip-articol.nume-câmp.*

Exemplu:

– referirea câmpului medie din articolul *elev* **e.medie**

– referirea câmpului absențe din articolul *elev* **e.absente**

– referirea unui element, i, al câmpului medii din articolul *elev* **e.medii[i]**

Prelucrarea acestor date începe cu introducerea și memorarea lor în zona de memorie rezervată la definirea variabilei de tip articol; pentru verificare, datele memorate vor fi afișate. Fiecare câmp poate fi prelucrat prin operații specifice tipului corespunzător (Tabelul 13).

Tabelul 13. **Prelucrarea articolelor**

PASCAL	C / C++
Exemplu pentru articolul elev	
1. Introducerea și afișarea datelor	

Program exemplu;
{ se defineste articolul elev }
Type elev=record
. . . . . .
medie1, medie2, medie_an:*real*;
absente:*byte*;
. . . . . .
end;
 	
{se defineste variabila cu tipul elev}
 	
var e: *elev*;
 	
begin
 	
{se introduc datele unui elev}
write ('medie1:'); readln (e.medie1);
write ('medie2:'); readln (e.medie2);
write ('absente:'); readln (e.absente);
. . . . . .
{se afiseaza datele elevului}
writeln ('medie1:', e.medie1);
writeln ('medie:2', e.medie2);
writeln ('absente:', e.absente);
 	
end.
	#include <iostream.h>
	void main ()
	{// se defineste articolul elev
	Typedef struct
	{
	. . . . . .
	float medie1, medie2, medie_an;
	unsigned:absente;
	. . . . . .
	} elev;
	
	// se defineste variabila cu tipul elev
	
	elev *e*;
	
	// se introduc datele unui elev
	cout<< " medie1:"; cin>> e.medie1;
	cout<< " medie2:"; cin>> e.medie2;
	cout<< " absente:"; cin>> e.absente;
	. . . . . .
	// se afiseaza datele elevului
	
	cout<< " medie1:"<<e.medie1<<endl;
	cout<< " medie2:"<<e.medie2<<endl;
	cout<< " absente:"<< e.absente<<endl;
	}

2. Operații asupra datelor	

{ se calculează media anuală}
e.medie_an := (e.medie1 +e.medie2)/2;
 	
{se afiseaza media anuala}
writeln('medie anuala: ', e.medie_an);
	// se calculează media anuală
	e.medie_an= (e.medie1 +e.medie2)/2;
	
	// se afiseaza media anuala
	cout<<:"medie anuala:"<< e.medie_an;

3. Introducerea și afișarea datelor în câmpuri grupate	

Type elev=record
. . . . . .
medii : array [1..3] of *real*;
. . . . . .
end;
 	
{se defineste variabila cu tipul elev}
var e: *elev*; i:byte;
begin
{se introduc mediile unui elev}
for i:=1 to 3 do
begin
write ('media:',i);
readln (e.medii[i]);
end;
end
	Typedef struct
	{
	. . . . . .
	float medii [3];
	. . . . . .
	} elev;
	// se defineste variabila cu tipul elev
	elev e;
	int i;
	{
	// se introduc mediile unui elev
	for (i=0; i<3;i++)
	{cout<< " media:"<<i;
	cin>> e.medii[i];
	}
	}

TEME

1. Descrieți sub formă de articol următoarele entități:
a) carte; b) mașină; c) calculator; d) persoană.

2. Definiți, în limbajul de programare studiat, tipurile de date și variabilele corespunzătoare articolelor descrise la cerința precedentă.

3. Realizați, în limbajul de programare studiat, un program care să determine trimestrul calendaristic al zilei curente. Se va folosi un articol cu câmpurile: *zi, lună, an*.
Exemplu: pentru data curentă: ziua 29 luna 08 anul 2007, se va afișa: trimestrul 3.

4. Realizați, în limbajul de programare studiat, un program care să determine dacă două puncte *A* și *B* îndeplinesc una dintre următoarele condiții:
a) punctele A și B se află pe o dreaptă paralelă cu axa OX;
b) punctele A și B se află pe o dreaptă paralelă cu axa OY;
c) punctele A și B se află pe o dreaptă egal depărtată de axele OX si OY.
Pentru fiecare punct se cunosc coordonatele *x* și *y*. Fiecare punct va fi descris printr-un articol.
Exemplu: Pentru punctul A de coordonate x=3, y=10 și punctul B de coordonate x=35, y=10 se va afișa mesajul: *punctele se află pe o dreaptă paralelă cu axa OX*.

6.4. Gruparea datelor organizate în structuri neomogene

Datele cu semnificații diferite care necesită gruparea în structuri neomogene reprezintă, în cele mai frecvente cazuri, caracteristici sau atribute ale unei situații din realitate (entitate): persoană, produs, carte, mașină etc. Memorarea datelor prin care poate fi descrisă o entitate se face într-o singură zonă de memorie (variabilă); dacă în problemă sunt prelucrate date despre mai multe *instanțe* ale unei *entități* (mai multe persoane, mai multe produse), se vor păstra întotdeauna datele ultimei instanțe introduse (ultima persoană, ultimul produs).

Exemplu:

La un magazin, se vând într-o zi ***n*** *produse. Se cunosc cantitatea și prețul fiecărui produs; se dorește calcularea și afișarea valorii totale a produselor vândute într-o zi.*

Rezolvarea problemei necesită introducerea, pe rând, a datelor despre fiecare produs. Pentru fiecare produs introdus, se calculează valoarea:

*valoare = cantitate * pret.*

Valoarea calculată se însumează la *total:*

total= total + valoare.

La sfârșit, se afișează *totalul.* (Tabelul 14)

Tabelul 14. **Exemplu** ***Magazin***

PASCAL	C/C++

program produse;
 	
{ se defineste articolul produs }
Type produs=record
cantitate:*byte*;
pret:*real*;
end;
{se defineste variabila cu tipul produs}
var p: *produs*;
{se definesc variabilele de lucru}
var valoare, total: real;
n, i :integer;
 	
begin
 	
{se introduce numarul de produse}
Write('numarul de produse:');
readln (n);
total:=0;
for i:= 1 to n do
begin
	#include <iostream.h>
	void main ()
	{
	// se defineste articolul produs
	Typedef struct
	{*unsigned* cantitate
	float pret;
	} produs;
	// se defineste variabila cu tipul produs
	produs p;
	//se definesc variabilele de lucru
	float valoare, total=0;
	
	int n, i ;
	
	//se introduce numarul de produse
	cout<< " numarul de produse:";
	cin>> n;
	for (i=1; 1<n; i++)
	{
	//se introduc datele unui produs

{se introduc datele unui produs}
write('cantitate:');
readln (p.cantitate);
write ('pret:'); readln (p.pret);
 	
{se calculeaza valoarea produsului}
valoare:= p.cantitate * p.pret;
{se insumează valoarea la total}
total:= total + valoare;
end;
 	
{se afiseaza valoarea totala}
Writeln ('valoarea totala:', total);
end
	cout<< " cantitate:";
	cin>>p.cantitate;
	cout<< " pret:"
	cin>>p.pret;
	
	//se calculeaza valoarea produsului
	valoare = p.cantitate * p.pret;
	//se insumează valoarea la total}
	total= total + valoare;
	}
	
	//se afiseaza valoarea totala
	cout<<"valoarea totala:"<<total;
	}

În cele mai multe situații reale, se păstrează datele fiecărei instanțe (fiecare persoană, fiecare produs) pentru prelucrări la nivel de grup: sortări, selecții. În *Catalogul clasei*, se păstrează datele despre toți elevii.

Gruparea instanțelor unei entități (toate persoanele, toate produsele, toți elevii) se face prin memorarea valorilor corespunzătoare în tablouri unidimensionale care, în acest caz, devin *vectori de articole*. Un element al vectorului reprezintă o instanță (o persoană, un produs, un elev) și este de tip articol.

Câmpul unui articol dintr-un vector de articole se adresează punctual, cu precizarea că, în acest caz, numele articolului este înlocuit cu adresa acestuia în vector (Tabelul 15).

Tabelul 15. **Vectori de articole**

PASCAL	C/C++

{definirea tipului de date articol}
Type articol=record
camp1:*tip1*;
camp2:*tip2*;
. . . . . .
end;
{definirea vectorului cu n elemente de tipul asociat articolului}
var vector:array [1..10] of articol;
var i : byte;
{adresarea unui camp dintr-un articol oarecare,i}
vector [i].camp
	// definirea tipului de date articol
	Typedef struct
	{ *tip1* camp1;
	tip2 camp2;
	. . . . . .
	} articol;
	// definirea vectorului cu n elemente
	//de tipul asociat articolului
	articol vector [10];
	int i;
	//adresarea unui camp dintr-un articol
	//oarecare,i}
	vector [i].camp

Exemplul 1 *prelucrarea a **n** produse*

PASCAL	C/C++

program produse;
{ se defineste articolul produs }
Type produs=record
cantitate:*byte*;
pret:*real*;
end;
 	
{se defineste vectorul p cu 100 de elemente de tip produs}
var p:array [1..100] of *produs*;
 	
{se definesc variabilele de lucru}
var valoare, total: real;
n, i :integer;
begin
 	
{se introduce numarul de produse}
write('numarul de produse:');
readln (n);
total:=0;
for i:= 1 to n do
begin
 	
{se introduc si se memoreaza datele fiecarui produs}
write('cantitate:');
readln (p[i].cantitate);
write ('pret:'); readln (p[i].pret);
 	
{se calculeaza valoarea produsului}
valoare:= p[i].cantitate * p[i].pret;
{se însumează valoarea la total}
total:= total + valoare;
end;
{se afiseaza valoarea totala}
writeln ('valoarea totala:', total);
end.
	#include <iostream.h>
	void main ()
	{
	// se defineste articolul produs
	Typedef struct
	{unsigned cantitate
	float pret;
	} produs;
	
	// se defineste vectorul p cu 100
	//elemente de tip produs
	produs p[100];
	
	//se definesc variabilele de lucru
	float valoare, total=0;
	int n, i ;
	
	//se introduce numarul de produse
	cout<< " numarul de produse:";
	cin>> n;
	for (i=0; i<n; i++)
	{
	
	//se introduc si se memoreaza datele //
	fiecarui produs
	cout<< " cantitate:";
	cin>>p[i].cantitate;
	cout<< " pret:"
	cin>>p[i].pret;
	
	//se calculeaza valoarea produsului
	valoare = p[i].cantitate * p[i].pret;
	{se însumează valoarea la total}
	total= total + valoare;
	}
	//se afiseaza valoarea totala
	cout<<"valoarea totala:"<<total;
	}

Exemplul 2 sortarea a ***n*** produse - metoda *Bubble sort*	

var aux: produs; s:byte;
begin
{sortare crescatoare dupa campul pret}
repeat
s:=0;
for i:=1 to n-1 do
begin
if p[i].pret> p[i+1].pret
then
{interschimbarea se face la nivel de articol }
begin
aux:=p[i];
p[i]:= p[i+1];
p[i+1]:= aux;
s:=1;
end;
until s=0;
writeln (' lista preturilor');
for i:=1 to n do
writeln (p[i].pret);
end
	{produs aux; unsigned s;
	//sortare crescatoare dupa campul pret
	do
	{
	s=0;
	for (i=0; i< n-1; i++)
	if p[i].pret> p[i+1].pret
	
	//interschimbarea se face la nivel de articol
	{
	aux=p[i];
	p[i]= p[i+1];
	p[i+1]= aux;
	s=1;
	}
	}
	while (s);
	cout<<"lista preturilor"<<endl;
	for (i=0; i< n; i++)
	cout<<p[i].pret<<endl;
	}

TEME

1. Formulați exemple care să necesite organizarea datelor în vectori de articole.

2. Prezentați o situație reală care să necesite ordonarea datelor organizate în vectori de articole.

3. Alcătuiți o listă cu prelucrări specifice datelor grupate în tablouri unidimensionale; pentru fiecare prelucrare, construiți câte un exemplu care să necesite gruparea articolelor în vectori de articole.

De reținut! ARTICOL

- **structură de date necesară pentru înregistrarea informațiilor despre un aspect al realității (obiect, persoană, activitate) cu mai multe caracteristici;**
- **fiecare caracteristică formează un câmp al articolului;**
- **fiecare câmp poate avea un tip propriu; acest aspect determină proprietatea de structură neomogenă;**
- **memorarea și prelucrarea datelor organizate în structuri neomegene se face la nivel de câmp;**
- **referirea unui câmp se face prin adresare punctuală:**
  - **nume_articol.nume_câmp;**
- **articolele pot fi grupate în structuri omogene: vectori de articole;**
- **referirea unui câmp de articol dintr-un vector se face prin adresare punctuală, înlocuindu-se numele articolului cu adresa acestuia în vector.**

PROBLEME PROPUSE

1. Colecție

Se dorește înregistrarea următoarelor date despre obiectele dintr-o colecție: denumire, anul achiziției, valoare.

Cerințe:

a) realizați un program pentru introducerea și afișarea datelor despre un obiect din colecție;

b) realizați un program pentru introducerea și afișarea datelor despre mai multe obiecte din colecție (max. 100);

c) realizați un program care să determine cel mai vechi obiect din colecție;

d) realizați un program care să afișeze obiectele din colecție în ordinea descrescătoare a vechimii acestora.

2. Concurs

Se dorește înregistrarea următoarelor date despre candidații înscriși la un concurs: numele, data nașterii (zi, lună, an), are carnet de conducere (da/nu).

Cerințe:

a) realizați un program pentru introducerea și afișarea datelor despre un singur candidat;

b) realizați un program pentru introducerea și afișarea datelor despre mai mulți candidați (max. 100);

c) realizați un program care să determine candidații cu aceeași vârstă, *v*, introdusă de la tastatură;

d) realizați un program care să afișeze, în ordinea vârstei, candidații cu carnet de conducere.

3. Catalogul clasei

Realizați un program cu meniu care să rezolve următoarele cerințe ale profesorului diriginte:

– înregistrarea datelor pentru fiecare elev;
– afișarea datelor despre elevi;
– determinarea și afișarea mediei generale a clasei la sfârșit de semestru;
– afișarea elevilor în ordine descrescătoare, după media generală semestrială;
– determinarea mediei generale, anuale, pentru fiecare elev;
– afișarea elevilor în ordine descrescătoare, după media generală anuală;
– determinarea și afișarea mediei generale a clasei la sfârșit de an;
– afișarea elevilor în ordine crescătoare, după numărul de absențe.

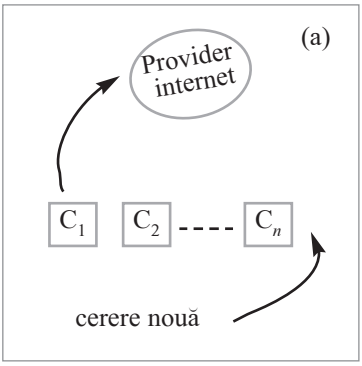
7. ORGANIZAREA DATELOR ÎN STRUCTURI DINAMICE

7.1. Modele de structuri dinamice

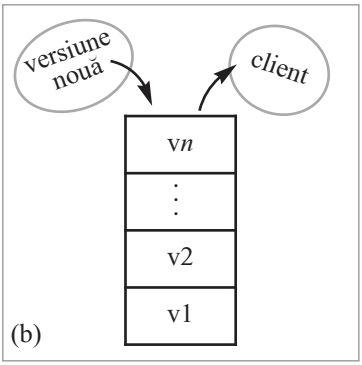
În foarte multe situații reale, există relații sau reguli care trebuie modelate astfel încât soluția de organizare a datelor să respecte atât semnificația, cât și restricțiile de comportament specifice.

Exemple:

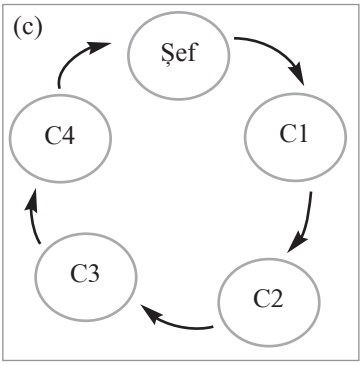
Exemplul 1. Persoanele care solicită un serviciu – conectarea la Internet printr-un provider autorizat – se înscriu pe o listă de aşteptare. O cerere nouă este aşezată, întotdeauna, ultima în listă. Serviciul de conectare este acordat, întotdeauna, primului solicitant din listă; după acordarea serviciului, cererea este eliminată din listă (fig. 20 a).



Exemplul 2. O firmă de software a realizat un program antivirus; programul este îmbunătățit continuu, prin tratarea de noi viruși; oferta de piață a firmei este organizată astfel încât clienții să aibă acces la program, începând întotdeauna cu ultima versiune a acestuia (fig. 20 b).



Exemplul 3. Pentru a comunica rapid și sigur, „șeful” unui grup de copii a întocmit o listă astfel încât un mesaj să poată fi transmis întregului grup, din copil în copil; „șeful” transmite mesajul primului copil din listă; ultimul copil comunică „șefului” că mesajul a ajuns la el (fig. 20 c).



4. Fiecare persoană are exact doi părinți; fiecare părinte este o persoană care la rândul ei are exact doi părinți. Pentru a păstra atât datele despre persoane, cât și relațiile directe de rudenie copil-părinți, se construiește arborele de familie – arborele genealogic (fig. 20 d).

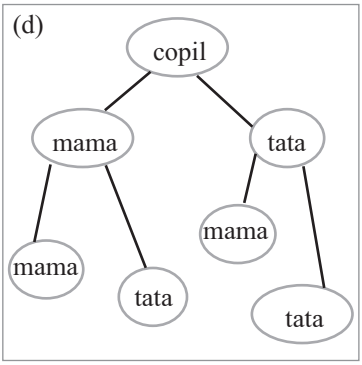


Figura 20

Fiecare dintre exemplele propuse necesită organizarea datelor după un model propriu: (1) *modelul firului de așteptare*; (2) *modelul stivei*; (3) *modelul listei circulare*; (4) *modelul arborescent*.

Implementarea acestor modele într-un limbaj de programare necesită soluționarea următoarelor probleme:

– organizarea datelor după semnificația acestora, cel mai frecvent în structuri omogene: tablouri unidimensionale;

– așezarea (intrarea) unui element din structură după regula specifică modelului;

– scoaterea (ieșirea) unui element din structură după regula specifică modelului;

– accesul la elementele structurii (numărarea/listarea elementelor) după regula specifică modelului.

În fiecare dintre situațiile reale din exemplele prezentate, numărul elementelor variază în timp: oricând poate să apară un solicitant pentru serviciul Internet sau o versiune nouă a programului antivirus; și în grupul de copii poate intra sau poate pleca un copil; în orice familie, copiii devin la rândul lor părinți, și arborele genealogic crește.

Întrucât numărul elementelor nu este constant și nici nu poate fi precizat în timp, structurile de date folosite pentru implementarea acestor modele se numesc *structuri dinamice*. Variația în timp a numărului de elemente (aspectul dinamic al structurii) respectă relațiile și disciplina (regulile de comportament) specifice modelului. Implementarea structurilor dinamice prin memorarea acestora în tablouri unidimensionale folosește *alocarea statică* a memoriei (mecanism de alocarea a memoriei din segmentul de date prin care zona de memorie maxim alocată – corespunzător capacității tabloului – rămâne la dispoziția programului pe toată durata de execuție a acestuia).

Aspectul dinamic al structurilor de date este pus și mai bine în evidență în modul de *alocare dinamică* a memoriei: mecanism de alocarea a memoriei de tip *Heap* prin care zonele de memorie pot fi solicitate și eliberate chiar în timpul execuției programului.

TEME

1. Formulați un exemplu real care să necesite organizarea datelor într-un model de tip *fir de așteptare*. Puneți în evidență aspectul dinamic al structurii.

2. Formulați un exemplu real care să necesite organizarea datelor într-un model de tip stivă. Puneți în evidență aspectul dinamic al structurii.

3. Formulați un exemplu real care să necesite organizarea datelor într-un model de tip *listă circulară*. Puneți în evidență aspectul dinamic al structurii.

4. Formulați un exemplu real care să necesite organizarea datelor într-un model de tip *arborescent*. Puneți în evidență aspectul dinamic al structurii.

5. Asociați modelul dinamic corespunzător fiecăreia dintre următoarele situații:

a) organizarea calculatoarelor într-o rețea locală de tip inel;

b) organizarea aplicațiilor deschise de un utilizator în sistemul de operare Windows;

c) organizarea informațiilor pe discul sistem;

d) organizarea cererilor de listare la imprimantă;

e) organizarea instrucțiunilor unui program.

ARBORELE GENEALOGIC
temă de compoziţie

Realizați arborele genealogic personal printr-o prezentare cât mai atractivă care să pună în evidența personalitatea fiecărui membru al familiei.

Se poate lucra în oricare dintre aplicațiile cu efecte grafice cunoscute.

Sugestie de rezolvare:

– prezentarea acestei teme de către elevi, în laborator, conduce la o activitate foarte atractivă; se pot face „clasamente”: cel mai „înalt” arbore; cel mai „vârstnic” arbore, cel mai „stufos” arbore.

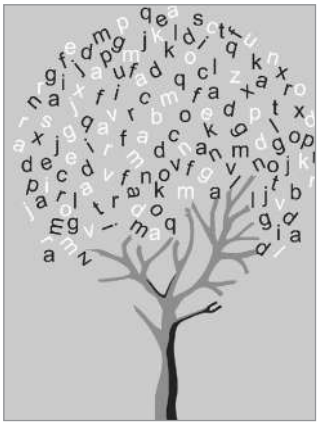
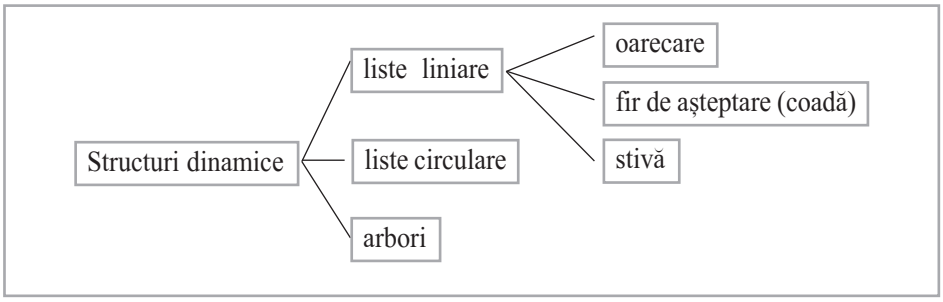


Figura 21

7.2. Clasificarea structurilor dinamice

Relațiile dintre elemente determină mai multe categorii de structuri dinamice prezentate în clasificarea din figura 22.

Figura 22. Clasificarea structurilor dinamice



Fiecare dintre categoriile de structuri dinamice din această clasificare are proprietăți specifice (Tabelul 16).

Tabelul 16. **Proprietățile structurilor dinamice**

STRUCTURA DINAMICĂ	PROPRIETĂȚI
Listă liniară	Relațiile dintre elementele structurii sunt de tip succesor – predecesor; există un singur element fără predecesor, capul listei, și un singur element fără succesor, ultimul element din listă.
Listă oarecare	Nu există nicio regulă pentru intrarea și ieșirea elementelor din structură.
Fir de așteptare	Intrarea și ieșirea elementelor din structură se face după regula FIFO (First Input First Output = *primul intrat, primul ieșit*).

Stivă	Intrarea și ieșirea elementelor din structură se face după regula LIFO (Last Input First Output = *ultimul intrat, primul ieșit*).
Listă circulară	Relațiile dintre elementele structurii sunt de tip succesor – predecesor; nu există niciun element fără predecesor sau fără succesor.
Arbori	Relațiile dintre elementele structurii sunt de tip ascendent – descendent; există un singur element fără ascendent – rădăcina arborelui – și unul sau mai multe elemente fără descendenți – elementele terminale sau frunze.

TEME

1. Identificați tipul listei liniare care poate fi asociat următoarelor situații:

a) Profesorul diriginte întocmește lista cu elevii care vor participa la o excursie cu număr limitat de locuri (mai puține decât efectivul clasei).

b) Întrucât numărul elevilor care doresc să meargă în excursie este mult mai mare decât numărul de locuri, profesorul diriginte reface lista pentru a-i elimina mai ușor pe cei care s-au înscris mai târziu.

c) La ora de Educație Fizică, elevii intră pe rând din vestiar în sala de sport; fiecare elev trebuie să se alinieze, ocupându-și locul astfel încât, în fiecare moment, șirul elevilor prezenți în sală să fie ordonat descrescător după înălțime.

2. Se consideră o listă liniară oarecare cu *n* elemente (n>10).

Determinați valoarea următoarelor expresii:

a) succesor (element 4) = b) predecesor (element 9) =
c) succesor (element n) = d) predecesor (element n) =
e) succesor (element 1) = f) predecesor (element 1) =
g) succesor (predecesor (n)) = h) predecesor (predecesor (3)) =

3. Stabiliți relația prin care o listă liniară oarecare cu *n* elemente poate fi transformată într-o listă circulară.

4. 4.1. Pentru fiecare membru din arborele genealogic personal, determinați:

a) numărul descendenților;

b) numărul ascendenților;

c) numărul elementelor terminale (fără descendenți).

4.2. Precizați care este semnificația elementelor terminale din arborele genealogic personal.

4.3. Care este semnificația elementului rădăcină din arborele genealogic personal?

4.4. Cum ar trebui construit arborele genealogic personal, astfel încât autorul să fie un element terminal?

7.3. Prelucrări specifice structurilor dinamice liniare

Pentru organizarea datelor în structuri dinamice liniare, sunt necesare următoarele prelucrări:

● **Crearea** structurii dinamice: această prelucrare corespunde memorării datelor pentru primul element din structură.

● **Parcurgerea** structurii dinamice: această prelucrare permite localizarea fiecărui element din structură, respectându-se regulile de ordine specifice modelului de structură dinamică.

● **Actualizarea** structurii dinamice: această prelucrare permite modificarea numărului de elemente din structură prin adăugare sau inserare de elemente noi sau prin eliminarea unor elemente; tot prin actualizare, se pot modifica informațiile specifice unui element.

În Tabelul 17 sunt descrise operațiile necesare implementării acestor prelucrări.

Tabelul 17. **Structuri dinamice – prelucrări și operații specifice**

<table>
<tr><th colspan="2" rowspan="2">PRELUCRARE</th><th rowspan="2">OPERAȚII SPECIFICE</th><th colspan="2">STAREA STRUCTURII
(numărul n de elemente)</th></tr>
<tr><th>ÎNAINTE DE PRELUCRARE</th><th>DUPĂ PRELUCRARE</th></tr>
<tr><td colspan="2">CREARE</td><td>– se verifică dacă structura este vidă
– se memorează datele pentru primul element din structură</td><td>dacă n=0
(structura este vidă)</td><td>atunci
n=1
altfel
operație fără sens</td></tr>
<tr><td rowspan="2">PARCURGERE</td><td>Totală</td><td>– localizarea tuturor elementelor</td><td colspan="2" rowspan="2">Nu se modifică nici numărul, nici valorile elementelor.</td></tr>
<tr><td>Parțială</td><td>– localizarea elementelor care îndeplinesc o condiție specificată</td></tr>
<tr><td rowspan="3">ACTUALIZARE</td><td>Adăugare / Inserare</td><td>– se verifică dacă s-a completat capacitatea structurii
– se memorează datele pentru un element nou care intră în structură respectând regula de intrare specifică modelului.</td><td>dacă
n=capacitatea structurii</td><td>atunci
operație imposibilă
altfel
n+1 elemente</td></tr>
<tr><td>Ștergere</td><td>– se verifică dacă structura este vidă
– se elimină din structură un element respectând regula de ieșire specifică modelului.</td><td>dacă
n= 0
structura este vidă</td><td>atunci
operație imposibilă
altfel
n-1 elemente</td></tr>
<tr><td>Modificare</td><td>Se localizează (prin parcurgere parțială) elementul ale cărui valori trebuie modificate; se modifică valorile elementului localizat.</td><td colspan="2">Nu se modifică numărul de elemente din structură; se modifică valorile pentru unul sau mai multe elemente.</td></tr>
</table>

TEME

1. Precizați de ce este necesară cunoașterea capacității unei structuri dinamice în varianta implementării prin alocarea statică a memoriei.

2. Identificați situațiile în care pot fi date următoarele mesaje:

a) *operație fără sens, listă existentă;*

b*) operație imposibilă, listă vidă;*

c) operație imposibilă, element inexistent;

d*) nu este permisă operația de inserare.*

3. Alcătuiți câte o secvență de operații elementare, necesară fiecăreia dintre următoarele prelucrări:

a) afișarea numărului de elemente dintr-o listă oarecare;

b) intrarea unui element nou într-un fir de așteptare;

c) afișarea numărului de ordine al elementelor care respectă o condiție specificată;

d) verificarea existenței în listă a unui element care respectă o condiție specificată;

e) ieșirea unui element dintr-o stivă.

7.4. Implementarea structurilor dinamice liniare

7.4.1. FIRUL DE AȘTEPTARE (COADA)

Disciplina structurii dinamice *fir de așteptare* sau *coadă* este de tip FIFO (First Input First Output = *primul intrat, primul ieșit*). Implementarea într-un limbaj de programare a acestui model de structură dinamică revine la controlul operațiilor de intrare/ieșire în/din structură, astfel încât să fie respectată disciplina FIFO.

În acest scop, structura va fi controlată prin:

(1) doi marcatori (indici) de poziție pe care îi vom numi *primul* și *ultimul* (*fig. 22).*

Figura 22

(2) *capacitatea structurii* (numărul maxim de elemente alocate pe care îl vom nota cu *n*).

Cazuri particulare

Dacă firul de așteptare nu conține niciun element, firul este gol (*fir vid*).

Dacă a fost ocupată toată capacitatea structurii, *firul este plin*.

Prelucrările specifice firului de așteptare

▸ **crearea** are sens doar dacă firul este gol (fir vid);

▸ **intrarea** unui element nou:

– un element poate intra în structură, doar dacă nu a fost completată capacitatea structurii („mai sunt locuri libere”);

– întotdeauna, noul element se așază la sfârșitul structurii, el devine *ultimul*;

▸ **ieșirea** unui element (întrucât firele de așteptare/coada se formează pentru satisfacerea unor cereri, „servicii”, spunem că ieșirea din fir are loc când *primul* element a fost *servit*):

– un element poate ieși din structură, doar dacă firul nu este vid (există cel puțin un element);

– întotdeauna, după ieșirea unui element din structură, toate elementele se deplasează „în față”; coada „avansează” și elementul ajuns pe poziția *primul* poate fi *servit (fig. 23);*

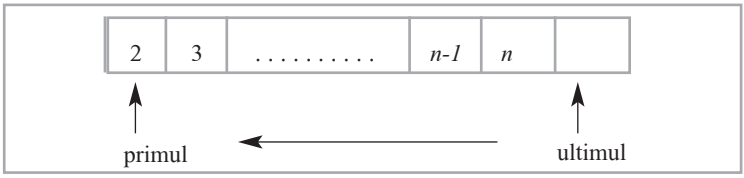


Figura 23

▸ **parcurgerea** se poate efectua doar dacă firul nu este vid:

– parcurgerea se poate face pentru determinarea numărului de elemente „așezate la coadă”, pentru afișarea elementelor sau pentru determinarea unor proprietăți;

– parcurgerea se face întotdeauna de la *primul* la *ultimul* element.

Elementele firului de așteptare pot fi persoane, obiecte, procese care așteaptă satisfacerea unor cereri; exemple:

– persoanele înscrise pe o listă de așteptare pentru rezervări de bilete, acceptarea la un interviu etc.;

– persoanele care așteaptă la rând pentru efecturea unor plăți, cumpărarea de produse etc.;

– mașinile aflate într-un spațiu de parcare la o benzinărie;

– evenimentele planificate într-o locație (concerte, concursuri etc.).

În cele mai frecvente situații, se înregistrează mai multe informații despre elementele firului de așteptare. Fiecare element din fir este descris ca o structură eterogenă – articol; firul de așteptare va fi memorat într-un vector de articole.

Pentru început, vom considera că fiecare element este descris prin numărul său de ordine (vector cu valori întregi, pozitive).

În continuare, se prezintă secvența pseudocod a operațiilor necesare implementării unui fir de așteptare.

```
begin fir _de_ aşteptare_1
// se lucrează  cu vectorul F pentru care se alocă 100 de elemente
//  variabile de lucru:
// n numărul maxim de elemente  din fir (capacitatea firului )
// primul    ultimul  marcatori de poziţie
//  i indicele de adresă pentru F
// nr  numărul de elemente din fir

// secvenţa de iniţializare
// secvenţa poate fi completată cu validarea lui n faţa de numărul de elemente alocate (100)
scrie introduceti valoare pentru capacitatea firului de asteptare, n=
citeşte n
pentru i=1 la n execută
F[i] ← 0
sfârşit pentru
 primul←0     // fir vid; se consideră că adresarea elementelor din vector începe de la 1
 ultimul←0
// sfârşit secvenţa de iniţializare

// secvenţa pentru creare
dacă  primul = 0
atunci
  bloc
            primul←1
            scrie introduceti valoare pentru primul element F[primul]=
            citeşte F[primul]
            ultimul← primul
   sfârşit  bloc
altfel
       scrie    operatie fara sens: firul nu este vid
sfârşit dacă
//sfârşit  secvenţa creare

// secvenţa pentru intrarea unui element nou
ultimul ← ultimul+1
dacă ultimul < =n
       atunci
                      bloc
             scrie introduceti valoare pentru noul  element F[ultimul]=
citeşte F[ultimul]
            sfârşit  bloc

           altfel
                        scrie    operatie imposibila:  firul este plin
sfârşit dacă
//sfârşit  secvenţa  pentru intrarea unui element nou
```

```
// secvența pentru ieșirea (servirea) unui element
dacă primul =o
      atunci
                        scrie    operatie imposibila:  firul este gol
          altfel
                  bloc
     scrie   este servit elementul F[primul]

//  coada avansează – secvența A
pentru i= primul  la ultimul-1 execută
          F [i] ← F[i+1]
 sfârșit   pentru
F[ultimul] ← 0
ultimul ← ultimul -1
                  sfârșit  bloc
sfârșit dacă
//sfârsit  secvența  pentru ieșirea unui element

secvența B
dacă primul < ultimul
    atunci
F[primul] ← 0
primul ← primul +1
sfârșit  dacă

//secvența pentru parcurgerea firului: se afișează elementele din șir și lungimea firului
dacă primul =o
      atunci
                        scrie    operatie imposibila:  firul este gol
          altfel
              bloc
                        nr ← 0
                        pentru i= primul  la ultimul execută
                                bloc
                                        scrie  F [i]
                                        nr ←  nr + 1
                                 sfârșit  bloc
                        sfârșit pentru
                       scrie  firul contine nr elemente
                 sfârșit  bloc
sfârșit dacă
//sfârșit  secvența  pentru  parcurgerea firului de așteptare

sfârșit fir _de_ așteptare_1
```

Pentru simularea aspectului dinamic al firului de aşteptare, sugerăm implementarea prelucrărilor specifice printr-un program cu meniu (fig. 24).

```
begin   fir _de_ aşteptare_2
//se afişează opţiunile din meniu
optiune ← 0
repetă
// se sterge ecranul
// afişare meniu
     repetă
scrie      fir de asteptare –prelucrări specifice
scrie  1 creare
scrie  2 intrarea unui  element nou
scrie  3 iesirea unui element
scrie  4 parcurgere
scrie   5 sfarsit program

scrie  introduceti optiunea (1,2,3,4,5):
citeste optiune
       până când  optiune <=5    // se validează optiunea
   selectează  optiune
optiune=1
                // secvenţă creare
optiune=2
               // secvenţă  intrare element nou
optiune=3
              // secvenţă ieşire element
optiune=4
             // secvenţă parcurgere
optiune=5
          scrie    sfarsit program
    sfârşit selectează
până când optiune=5

sfârşit fir _de_ aşteptare_2
```

Meniu *FIFO*

1. Creare
2. Intrare
3. Servire
4. Parcurgere
5. Exit

Optiune _

Figura 24

Printr-o singură rulare a programului, utilizatorul poate să aleagă din meniu, de mai multe ori, opţiunea corespunzătoare simulării firului de aşteptare.

Spre exemplu, pentru simularea intrării primului element urmată de intrarea a încă trei elemente şi ieşirea a două elemente, utilizatorul poate dirija execuţia programului prin următoarea secvenţă de opţiuni: 1, 4, 2, 4, 2, 4, 2 ,4, 3, 4, 3, 4, 5.

După fiecare opțiune care simulează dinamica structurii (creare, intrare, ieșire), s-a ales parcurgerea (opțiunea 4) prin care se afișează elementele din fir. În acest mod, utilizatorul poate controla dacă programul modelează corect firul de așteptare.

TEME

1. Codificați, în limbajul de programare studiat, fiecare dintre secvențele pseudocod propuse pentru simularea unui fir de așteptare.

2. Precizați care este efectul următoarei secvențe de opțiuni :
a) crearea și parcurgerea firului de așteptare;
b) crearea, intrarea unui nou element și parcurgerea firului de așteptare;
c) crearea, ieșrea unui element și parcurgerea firului de așteptare;
d) ieșirea unui element din firul de așteptare.

3. Care este efectul eliminării din program a secvenței de instrucțiuni pentru crearea firului de așteptare?

4. Compuneți un program nou care să nu conțină secvența de creare.

5. Pentru ieșirea unui element din coadă, au fost propuse două secvențe care simulează „avansul”: secvența A și secvența B. Urmăriți secvența B și răspundeți la următoarele întrebări:
a) Care este semnificația momentului *primul* = *ultimul*?
b) Cum pot fi utilizate locațiile de memorie eliberate prin ieșirea unui element?

6. Realizați un program cu meniu pentru modelarea următoarei situații:
Mai multe persoane se înscriu pentru rezervarea de bilete la un spectacol la care au acces doar elevii. Se cunosc numărul de bilete și anul nașterii fiecărei persoane înscrisă pe listă. Organizatorii doresc să afle numărul de persoane înscrise, câți elevi s-au înscris, câte bilete disponibile mai sunt.

7. Propuneți o situație reală a cărei rezolvare cu calculatorul să necesite modelarea datelor prin fire de așteptare.

7.4.2. STIVA

Disciplina structurii dinamice numită *stivă* este de tip LIFO (Last Input First Output =*ultimul intrat, primul ieșit*). Implementarea într-un limbaj de programare a acestui model de structură dinamică revine la controlul operațiilor de intrare/ieșire în/din structură, astfel încât să fie respectată disciplina LIFO.

În acest scop, structura va fi controlată prin:

♦ doi marcatori (indici) de poziție pe care îi vom numi *bază* și *vârf* (fig. 25);

♦ *capacitatea structurii* (numărul maxim de elemente alocate pe care îl vom nota cu *n*).

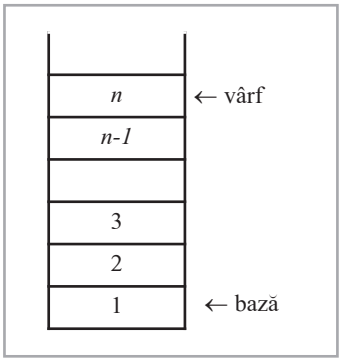


Figura 25

Cazuri particulare

- Dacă stiva nu conține niciun element, stiva este goală (*stivă vidă*).
- Dacă a fost ocupată toată capacitatea structurii, *stiva este plină*.

Prelucrările specifice stivei

▸ **crearea** are sens doar dacă stiva este goală (stivă vidă);

▸ **intrarea** unui element nou:

– un element poate intra în structură doar dacă nu a fost completată capacitatea structurii („mai sunt locuri libere”);

– întotdeauna, noul element se așază peste celelalte elemente, în vârful stivei;

– după intrarea unui element nou în stivă, spunem că „stiva crește”;

▸ **ieșirea** unui element:

– un element poate ieși din structură, doar dacă stiva nu este vidă (există cel puțin un element);

– întotdeauna, iese din structură elementul aflat în vârful stivei;

– după ieșirea unui element din stivă, spunem că „stiva scade ”;

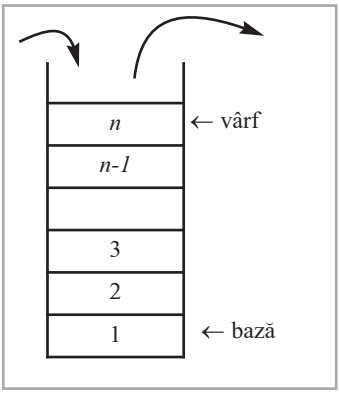


Figura 26

▸ **parcurgerea** se poate efectua doar dacă stiva nu este vidă:

– parcurgerea se poate face pentru determinarea numărului de elemente „stivuite”, pentru afișarea elementelor sau pentru determinarea unor proprietăți;

– parcurgerea se face întotdeauna de la *vârf* spre *bază*.

Elementele stivei pot fi persoane, obiecte, procese care așteaptă rezolvarea unei solicitări după regula LIFO; exemple:

– persoanele propuse pentru disponibilizare de la un loc de muncă al cărui manager ține seama de vechimea angajaților;

– vagoanele de tren aflate în probă pe o linie de manevră;

– programele activate de un utilizator *Windows*.

În cele mai frecvente situații, se înregistrează mai multe informații despre elementele aflate în stivă. Fiecare element din stivă este descris ca o structură eterogenă – articol; pentru memorarea elementelor din stivă se va folosi un vector de articole.

Pentru început, vom considera că fiecare element este descris prin numărul său de ordine (vector cu valori întregi, pozitive). În continuare, se prezintă secvența pseudocod a operațiilor necesare implementării stivei.

```
begin stiva1
// se lucrează cu vectorul S pentru care se alocă 100 de elemente
//  variabile de lucru:
// n numărul maxim de elemente din stivă (capacitatea stivei )
// baza  vârf  marcatori de poziție
//  i indicele de adresă pentru S
// nr  numărul de elemente din stivă

// secvența de inițializare
// secvența poate fi completată cu validarea lui n față de numărul de elemente alocate (100)

scrie introduceti valoare pentru capacitatea stivei, n=
citește n
pentru i=1 la n execută
S[i] ← 0
sfârșit pentru
 //   stiva  este vidă; se consideră că adresarea elementelor din vector începe de la 1
vârf←0
 baza←0
//  sfârșit secvența de inițializare

// secvența pentru creare
dacă  vârf = 0
atunci
  bloc
            vârf←1
scrie introduceti valoare pentru elementul din varful stivei S[vârf]=
citește S[vârf]
   sfârșit bloc
altfel
        scrie    operatie fara sens:stiva  nu este vida
sfârșit dacă
//sfârșit  secvența creare
```

```
// secvența pentru intrarea unui element nou
vârf ← vârf+1   //  stiva crește
dacă vârf < =n
      atunci
                bloc
            scrie introduceti valoare pentru elementul din varful stivei S[vârf]=
             citește S[vârf]
            sfârșit bloc

          altfel
                        scrie    operatie imposibila: stiva este plină
sfârșit dacă
//sfârșit  secvența  pentru intrarea unui element nou
// secvența pentru ieșirea unui element
dacă vârf =0
      atunci
                        scrie    operatie imposibila: stiva  este goala
          altfel
                 bloc
  scrie   iese elementul din varful stivei S[varf]
                       //  stiva scade
S[varf] ←0
varf ← varf – 1
                sfârșit bloc
sfârșit dacă
//sfârșit  secvența  pentru ieșirea unui element

//secvența pentru parcurgerea stivei:
//se afișează elementele din stivă și  înălțimea stivei
dacă vârf =o
      atunci
                        scrie    operatie imposibila: stiva este goala
          altfel
              bloc
                        nr ← 0
                        pentru i= vârf  la baza  execută
                                bloc
                                     scrie  S [i]
                                      nr ← nr + 1
                                 sfârșit  bloc
                        sfârșit pentru
                       scrie  stiva  contine nr elemente
                sfârșit  bloc
sfârșit dacă
//sfârșit  secvența  pentru  parcurgerea stivei
sfârșit stiva1
```

Pentru simularea aspectului dinamic al stivei, sugerăm implementarea prelucrărilor specifice printr-un program cu meniu (fig. 27).

```
begput   stiva2
//se afișează opțiunile din meniu
optiune←0

repetă
// se șterge ecranul
// afișare meniu
     repetă
scrie    operatii asupra stivei
scrie  1 creare
scrie  2 intrare element nou
scrie  3 iesirea unui element
scrie  4 parcurgere
scrie   5 sfarsit program

scrie  introduceti optiunea (1,2,3,4,5):
citește optiune
       până când  optiune <=5    // se validează optiunea

   selectează  optiune
optiune=1
                // secventa creare
optiune=2
               // secventa  intrare element nou
optiune=3
              // secventa iesire element
optiune=4
             // secventa parcurgere
optiune=5
          scrie    sfarsit program
    sfârșit selectează
până când optiune=5
sfârșit stiva2
```

Meniu *LIFO*

1. Creare
2. Intrare
3. Ieșire
4. Parcurgere
5. Exit

Optiune: _

Figura 27

- Printr-o singură rulare a programului, utilizatorul poate să aleagă din meniu, de mai multe ori, opțiunea corespunzătoare simulării stivei.
- Spre exemplu, pentru simularea intrării primului element urmată de intrarea a încă trei elemente și ieșirea a două elemente, utilizatorul poate dirija execuția programului prin următoarea secvență de opțiuni: 1, 4, 2, 4, 2, 4, 2, 4, 3, 4, 3, 4, 5.

• După fiecare opțiune care simulează dinamica structurii (creare, intrare, ieșire) s-a ales parcurgerea (opțiunea 4) prin care se afișează elementele stivei. În acest mod, utilizatorul poate controla dacă programul modelează corect stiva.

TEME

1. Codificați, în limbajul de programare studiat, fiecare dintre secvențele pseudocod propuse pentru simularea stivei.

2. Precizați care este efectul următoarei secvențe de opțiuni:
a) crearea și parcurgerea stivei;
b) crearea, intrarea unui nou element și parcurgerea stivei;
c) crearea, ieșirea unui element și parcurgerea stivei;
d) ieșirea unui element din stivă.

3. Precizați care este efectul eliminării din program a secvenței de instrucțiuni pentru crearea stivei.

4. Compuneți un program nou care să nu conțină secvența de creare a stivei.

5. Care este rolul marcatorului de poziție *baza* stivei?

6. Compuneți un program nou în care marcatorul *baza* să aibă valoarea *n* (capacitatea stivei).

7. Realizați un program cu meniu pentru modelarea următoarei situații:
La biblioteca școlii, cărțile de Informatică sunt atât de solicitate, încât bibliotecara nu le mai pune la loc în raft decât la sfârșitul zilei; în rest, le așază una peste alta pe masă; s-a constatat că nici elevii nu selectează cărțile și împrumută de fiecare dată cartea aflată „la vedere” (deasupra celorlalte cărți). Teancul de cărți nu poate depăși *n* exemplare ($n \leq 25$). Se cunoaște autorul, anul editării și prețul fiecărei cărți. Dorim să aflăm, la cerere, numărul cărților, valoarea acestora precum și numărul cărților împrumutate într-o zi. (Execuția programului va simula activitatea de restituire și împrumut pe parcursul unei zile.)

8. Propuneți o situație reală a cărei rezolvare cu calculatorul să necesite modelarea datelor prin structuri dinamice de tip stivă.

PROBLEME PROPUSE

1. La un cabinet medical, pacienții intră la consultație în ordinea sosirii. Despre fiecare pacient se cunoaște anul nașterii, genul (M/F), înălțimea și greutatea. Realizați un program care să simuleze înregistrarea pacienților în lista de așteptare, modificarea listei după fiecare consultație și determinarea următoarelor informații:
a) numărul pacienților care așteaptă pentru consultație;
b) numărul pacienților bărbați care așteaptă pentru consultație;
c) numărul pacienților supraponderali care așteaptă pentru consultație (greutatea ≥ 80 Kg și înălțimea ≤1,70m).

2. Pentru a rezolva mai repede solicitările unor clienți nemulțumiți de serviciile primite, *Oficiul pentru protecția consumatorilor* a decis să înființeze, pe lângă ghișeul de lucru *G1*, un ghișeu nou, *G2*. Coada formată la ghișeul *G1* este reorganizată astfel: clienții cu număr de ordine *par* trec la ghișeul *G2*.

Realizați un program pentru simularea celor două cozi: *G1* și *G2* pornind de la o coadă inițială, *G1*. Stabiliți singuri datele memorate pentru fiecare client.

3. Realizați un program pentru adunarea a două numere foarte mari. Justificați structura dinamică folosită.

4. Realizați un program pentru afișarea în ordine inversă a unui text introdus de la tastatură (exemplu: textul introdus este *mai mult ca perfectul* textul afișat este *lutcefrep ac tlum iam*). Justificați structura dinamică folosită.

5. Realizați un program pentru transformarea unui număr din baza 10 în baza 2. Justificați structura dinamică folosită.

6. Se consideră formate două cozi C1și C2; în cele două cozi sunt memorate valori numerice. Să se formeze o coadă nouă, C3, prin așezarea elementelor din coada C2 după elementele cozii C1.

Exemplu:
Coada C1 este: 7, 3, 6, 5, 8, 9, 2.
Coada C2 este: 6, 8, 1, 2, 5, 1, 3, 5, 9.
Se obține coada C3: 7, 3, 6, 5, 8, 9, 2, 6, 8, 1, 2, 5, 1, 3, 5, 9.

7. Se consideră formate două stive S1 și S2; în cele două stive sunt memorate valori numerice. Să se formeze o stivă nouă, S3, prin așezarea elementelor din stiva S2 peste elementele stivei S1.

Exemplu:
Stiva S1 este: 7, 3, 6, 5, 8, 9, 2.
Stiva S2 este: 6, 8, 1, 2, 5, 1, 3, 5, 9.
Se obține stiva S3: 6, 8, 1, 2, 5, 1, 3, 5, 9, 7, 3, 6, 5, 8, 9, 2.

8. Realizați un program care să prelucreze o secvență de comenzi pentru un calculator de buzunar; prima comandă executată este prima comandă din secvență. Fiecare comandă are următoarea structură:

operator1 operand operator2

unde: *operator1, operator2* sunt valori numerice reale iar *operand* poate avea valorile *S*-sumă, *D*–diferență *P*–produs, *E*–sfârșitul secvenței de comenzi (*sfârșit prelucrare*).

Exemplu:

Secvența de comenzi	rezultate
3.4 S 3	6.4
25.5 D 20.5	5
12 P 4	48
E	*sfârșit prelucrare*

9. Un șir de caractere conține duplicate în serie (caractere de același fel care se repetă unul după altul). Se dorește rafinarea șirului prin eliminarea duplicatelor în serie. Șirul vid este șir rafinat.

Exemplu:
Șirul inițial este: 912222223444355aa16777777800095.
Șirul final este: 96895.

10. Într-un grup de *n* copii există relații de prietenie; fiecare copil este identificat prin numărul său de ordine în grup; fiecare copil poate fi prieten cu mai mulți copii din grup. Profesorul diriginte dorește să așeze copiii într-o ordine care să pună în evidență relațiile de prietenie, și anume: începând cu un copil oarecare *i*, acesta își strigă toți prietenii – în ordinea crescătoare a numărului; prietenii copilului *i* se așază în rând, unul după altul. Primul copil care urmează în șir după copilul *i* își strigă toți prietenii – în ordinea crescătoare a numărului; aceștia se așază și ei în rând, unul după altul (dacă un copil a fost strigat înainte, el își păstrează locul în rând).

Exemplul 1: în grup sunt 6 copii.
Relațiile de prietenie dintre copii:

Figura 28

a)

	1		1		1
1		1		1	
	1		1	1	
1		1			
	1	1			
1					

Pentru i=2, ordinea copiilor în șir este: 2, 1, 3, 5, 4, 6.

Exemplul 2: în grup sunt 6 copii.
Relațiile de prietenie dintre copii:

b)

	1				1
1		1			
	1			1	
		1	1		
1					

Pentru i=3, ordinea copiilor în șir este: 3, 2, 5, 1, 4, 6.

Cerințe

a) Specificați structurile necesare organizării datelor din această problemă.
b) Determinați situațiile particulare care pot fi întâlnite în grupul de copii.
c) Formulați exemple numerice care să pună în evidență situațiile particulare determinate.
d) Realizați un program care să afișeze aranjarea copiilor din grup în ordinea dorită de profesorul diriginte.

MINIPROIECT ÎN ECHIPĂ. COMPANIA EFICIENT

Etapa: *Analiză* (identificarea datelor și a prelucrărilor)
Cerințe:
1. Analizați datele și cerințele prezentate în studiul de caz *Compania Eficient* (pag. 3); justificați formele de organizare a datelor prin analiza comparativă după eficiența prelucrării acestora cu calculatorul.
2. Alcătuiți documentația de proiect corespunzătoare acestei etape.