

Tablouri multidimensionale

Capitolul 13

- ❖ Aplicații cu tablouri tridimensionale
- ❖ Implementări sugerate
- ❖ Probleme propuse
- ❖ Soluțiile problemelor

Am văzut că într-un tablou unidimensional se pot păstra mediile generale ale elevilor unei clase (*MedGen*), nota unui elev fiind referită prin construcția *MedGen[nr]*, unde *nr* este numărul de ordine din catalog al elevului respectiv.

MedGen

numere de ordine →	1	2	...	30
medii →	9.33	8.20	...	9.63

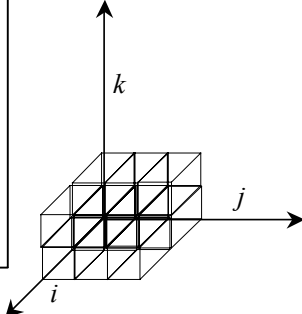
Mediile pe discipline ale fiecărui elev se pot păstra într-un tablou bidimensional.

	indici de coloană		1 (Limba rom.)	2 (Matem.)	...	12 (Educ. fizică)
indici de linie	1	Albu	7.33	9.20	...	10
	2	Ban	6.67	7.33	...	10

	30	Zeciu	9.10	10

Mediile la fiecare disciplină pe toți cei patru ani de liceu pentru toți absolvenții unei clase se vor codifica într-un tablou multidimensional, astfel:

indice <i>k</i>		An de studiu: 4				
		An de studiu: 3				
		An de studiu: 2				
		An de studiu: 1				
		indice <i>j</i>				
medie		Lb. rom.	Matem.	...	Ed. fiz.	
indice <i>i</i>		1	2	...	12	
Albu	1	7.33	9.20	...	10	
Ban	2	6.66	7.33	...	10	
...		
Zeciu	30	9.10	8.33	...	10	



Vom numi acest tablou *registru*. Un tablou tridimensional se reprezintă în spațiu printr-un paralelipiped dreptunghic, accesul la un element al său necesitând precizarea valorilor a trei indici (câte unul pentru fiecare dimensiune).

În tabloul prezentat ca exemplu, valoarea unui element este o medie și este localizat cu trei indici:

- primul indică elevul (număr de ordine din catalog);
- al doilea indice precizează materia de învățământ la care a fost obținută această medie (numărul de ordine al disciplinei);
- al treilea indice corespunde anului de studiu în care elevul a obținut acea medie la disciplina precizată.

De exemplu, *registru*[2,1,3] reprezintă media obținută de elevul Ban la Limba și literatura română în cel de-al treilea an de liceu.

În mod similar se pot imagina situații a căror reprezentare se pretează la structuri cu 4, 5 sau mai multe dimensiuni. Generalizând, putem spune că într-un tablou n -dimensional un element se regăsește precizând numele tabloului precum și valorile celor n indici.

13.1. Aplicații cu tablouri tridimensionale

13.1.1. Parcurgerea unei fețe a unui tablou tridimensional

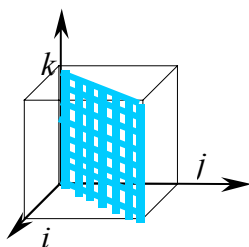
Considerăm problema în care trebuie să calculăm suma elementelor din tabloul tridimensional de dimensiuni m, n, p , aflate pe fața inferioară a paralelipipedului corespunzător tabloului. Elementele care fac parte din zona menționată au valoarea celui de-al treilea indice $k = 1$. Subalgoritmul care descrie calculul acestei sume este:

Subalgoritm Suma_Față(a, m, n):

```
Suma ← 0
pentru  $i=1, m$  execută:
  pentru  $j=1, n$  execută:
    Suma ← Suma +  $a[i, j, 1]$ 
  sfârșit pentru
sfârșit pentru
sfârșit subalgoritm
```

13.1.2. Parcurgerea unei secțiuni

Vom prezenta modalitatea în care se afișează elementele tabloului tridimensional a de dimensiuni $m \times m \times m$, aflate în tabloul bidimensional mărginit de elementele de forma $a[1,1,k]$ și elementele de forma $a[m,m,k]$ (suprafața formată din toate diagonalele principale ale tablourilor bidimensionale de coordonate i, j).



Acest tablou bidimensional se va afișa pe linii, de sus în jos.

```

Subalgoritm Secțiune(a,m,n) :
  pentru k=m,1 execută:
    pentru i=1,m execută:
      scrie a[i,i,k]
    sfârșit pentru
  sfârșit pentru
sfârșit subalgoritm

```

13.1.3. Parcurgerea tabloului tridimensional pornind de la fețele exterioare spre interior

Vom genera un tablou tridimensional cubic de dimensiune n , astfel încât pe fețele exterioare să conțină numai valori 1, cu o poziție mai în interior să conțină numai valori 2 și așa mai departe. La fiecare nivel, odată cu fiecare element al unei fețe se vor inițializa și celelalte cinci elemente corespunzătoare celorlalte fețe ale cubului. Se determină mai întâi numărul nivelurilor de adâncime pentru care trebuie generate fețe.

Exemplu

$n = 4$:

	1	2	3	4
1				
2				
3				
4				

$n = 5$:

	1	2	3	4	5
1					
2					
3					
4					
5					

Se observă că pentru $n = 4$ numărul de niveluri este 2 ($\lfloor n/2 \rfloor$). Atunci când numărul n este impar, numărul de niveluri de parcurs este $\lfloor n/2 \rfloor + 1$ (pentru $n = 5$ este 3). Pentru a unifica ambele situații într-o singură formulă vom parcurge $\lfloor n/2 \rfloor + \text{rest}[n/2]$ niveluri.

Subalgoritm Generare(a,n,nivel) :

```

  pentru i=nivel,n-nivel+1 execută:
    pentru j=nivel,n-nivel+1 execută:
      a[nivel,i,j] ← nivel
      a[i,j,nivel] ← nivel
      a[i,nivel,j] ← nivel
      a[n-nivel+1,i,j] ← nivel
      a[i,j,n-nivel+1] ← nivel
      a[i,n-nivel+1,j] ← nivel
    sfârșit pentru
  sfârșit pentru
sfârșit subalgoritm

```

```

Algoritm Generare_tablou_tridimensional:
  citește n
  pentru nivel=1, [n/2] + rest[n/2] execută:
    Generare(a,n,k)
  sfârșit pentru
sfârșit algoritm

```

13.1.4. Adunarea a două tablouri tridimensionale

Adunarea a două tablouri tridimensionale se face similar adunării a două tablouri bidimensionale. Singurul element de noutate este apariția unei dimensiuni în plus.

Dacă a și b sunt două tablouri tridimensionale de dimensiuni m , n și p , suma $a + b$ se calculează astfel:

```

Subalgoritm Adună(a,b,c):
  pentru i=1,m execută:
    pentru j=1,n execută:
      pentru k=1,p execută:
        c[i,j,k] ← a[i,j,k] + b[i,j,k]
      sfârșit pentru
    sfârșit pentru
  sfârșit pentru
sfârșit algoritm

```

13.1.5. Afișarea elementelor unui tablou tridimensional

Afișarea elementelor unui astfel de tablou poate fi făcută pe felii. Un exemplu de astfel de afișare este dat de următorul subalgoritm:

```

Subalgoritm Afișare(a,n):
  pentru i=1,n execută:
    pentru j=1,n execută:
      pentru k=1,n execută:
        scrie a[i,j,k]
      sfârșit pentru
    sfârșit pentru
  sfârșit pentru
sfârșit subalgoritm

```

Structura de date de tip tablou are numeroase aplicații în practică, dar majoritatea metodelor larg utilizate se concentrează pe tablouri uni și bidimensionale. În anumite situații metoda programării dinamice^{*)} necesită tablouri auxiliare tridimensionale pentru a păstra informații utile în construirea rezultatului.

^{*)} Metoda programării dinamice se va învăța în clasa a X-a.

13.2. Implementări sugerate

Pentru a vă familiariza cu modul de lucru cu tablouri de dimensiuni mai mari decât doi, vă propunem să realizați implementarea exercițiilor:

1. suma elementelor unui tablou n -dimensional;
2. identificarea coordonatelor unui element dat (toate aparițiile) într-un tablou n -dimensional;
3. parcurgerea unui tablou tridimensional dinspre exterior înspre interior;
4. determinarea matricei dintr-un tablou tridimensional care are suma elementelor maximă.

13.3. Probleme propuse

13.3.1. Populație virusată

Se urmărește evoluția îmbolnăvirii unei populații cu un virus, pe parcursul a p zile. Inițial populația este dată într-un tablou bidimensional. Referitor la vindecarea indivizilor bolnavi sau îmbolnăvirea celor sănătoși funcționează următoarele reguli:

- Un individ sănătos având cel puțin trei vecini infestați în jur, se îmbolnăvește și el.
- Un individ bolnav înconjurat numai de bolnavi se mobilizează și se vindecă.
- Orice individ bolnav pe durata a cel puțin 30 de zile se însănătoșește.

Prin vecini se înțeleg numai indivizii aflați în imediata apropiere pe aceeași linie sau pe aceeași coloană cu un anumit element considerat.

- Să se genereze tabloul tridimensional cuprinzând evoluția populației în cele p zile.
- Să se determine ziua sau zilele în care numărul indivizilor bolnavi a fost maximă.
- Să se afișeze coordonatele indivizilor care au fost bolnavi pe întreaga perioadă de p zile.

Date de intrare

Pe prima linie a fișierului **VIRUS.IN** se află dimensiunile tabloului bidimensional (m , n) urmate de numărul de zile (p) pe care se efectuează studiul. Pe următoarele m linii se află elementele tabloului inițial cuprinzând starea indivizilor în prima zi. Tabloul conține valori 0 și 1, 0 având semnificația unui individ sănătos, iar 1 însemnând un individ atins de virus.

Date de ieșire

Pe prima linie a fișierului **VIRUS.OUT** se vor scrie zilele în care numărul îmbolnăvirilor a fost maxim. Pe următoarele linii se vor scrie perechi de numere naturale reprezentând coordonatele acelor indivizi care au fost bolnavi în toată perioada celor p zile. Toate numerele scrise pe aceeași linie se vor separa prin câte un spațiu.

Restricții și precizări

- $1 \leq m, n \leq 10$;
- $1 \leq p \leq 50$.

Exemplu

VIRUS . IN	VIRUS . OUT		<i>Explicație</i>
5 5 5	2 5	3 2	Fișierul de ieșire este vizualizat pe două coloane pe motiv de spațiu.
1 0 1 0 1	1 1	3 4	
0 1 0 1 0	1 3	4 1	
0 1 0 1 0	1 5	4 3	
1 0 1 0 1	2 2	4 5	
0 1 0 1 0	2 4	5 2	
		5 4	

13.4. Soluțiile problemelor propuse**13.4.1. Populație virusată**

În această aplicație tabloul tridimensional este definit ca șir de tablouri bidimensionale astfel încât fiecărei zile să-i corespundă câte un tablou bidimensional privind starea sănătății din ziua respectivă.

Algoritmul de rezolvare conține subalgoritmi pentru rezolvarea subproblemelor:

1. Construirea tabloului tridimensional, pornind de la tabloul bidimensional dat, pe baza regulilor de transmitere a bolii sau de stopare a ei. Primul pas în abordarea acestei subprobleme este citirea dimensiunilor tabloului tridimensional care se va construi și a tabloului cu informațiile privind starea de sănătate din prima zi.

```

Subalgoritm Prima_zi(m,n,p,a):
    citește m,n,p                                { dimensiunile tabloului tridimensional }
    pentru i=1,n execută:
        pentru j=1,n execută:
            citește a[1,i,j]                      { citirea tabloului corespunzător primei zile }
        sfârșit pentru
    sfârșit pentru
sfârșit subalgoritm

```

Generarea tabloului tridimensional se realizează construind tablourile bidimensionale, corespunzătoare zilei, pornind din situația corespunzătoare zilei precedente.

```

Subalgoritm Generează(a,m,n): { pentru a ne putea referi și la vecinii unui }
                                     { element aflat pe margine, bordăm matricea inițială cu 0 }

    pentru i=0,m+1 execută:
        a[i,0,1] ← 0
        a[i,n+1,1] ← 0
    sfârșit pentru
    pentru j=0,n+1 execută:
        a[0,j,1] ← 0
        a[m+1,j,1] ← 0
    sfârșit pentru
    { generarea propriu-zisă }
    pentru zi=2,p execută: { pentru fiecare zi }
        pentru i=1,m execută:
            pentru j=1,n execută: { pentru fiecare element al matricei din ziua zi }
                                     { se numără vecinii virusați din ziua precedentă }
                număr ← a[zi-1,i-1,j] + a[zi-1,i+1,j] + a[zi-1,i,j-1] +
                                     a[zi-1,i,j+1];
                { a. dacă nu era virusat, dar are cel puțin trei vecini virusați se virusează și el }
                dacă a[zi-1,i,j] = 0 atunci
                    dacă număr ≥ 3 atunci a[zi,i,j] ← 1
                    altfel a[zi,i,j] ← 0
                sfârșit dacă
            { b. dacă era virusat și este înconjurat din toate părțile de viruși se însănătoșește }
            altfel { era virusat: a[zi-1,i,j] = 1 }
                dacă număr = 4 atunci
                    a[zi,i,j] ← 0
                altfel
                    a[zi,i,j] ← 1
                sfârșit dacă
            sfârșit dacă
        { c. dacă era virusat de 30 de zile, se însănătoșește }
        dacă (zi ≥ 30) și (a[zi-1,i,j] = 1) și
            (număr_de_zile(a,i,j,zi-1) = 30) atunci
            a[zi,i,j] ← 0
        sfârșit dacă
    sfârșit pentru
    sfârșit pentru
    sfârșit subalgoritm

```

Acest subalgoritm apelează un alt subalgoritm care, pornind de la ziua z_i , calculează de câte zile persoana identificată prin coordonatele i, j este bolnavă. Prezentăm în continuare acest subalgoritm.

```

Subalgoritm Număr_de_zile( $a, i, j, z_i$ ):
    { returnează numărul de zile de boală continuă ale persoanei de pe poziția  $i, j$  }
     $k \leftarrow z_i - 1$ 
    cât timp ( $k > 1$ ) și ( $a[k, i, j] = 1$ ) execută:
         $k \leftarrow k - 1$ 
    sfârșit cât timp
    număr_de_zile  $\leftarrow z_i - k + 1$ 
sfârșit subalgoritm

```

2. Rezolvarea primei cerințe din enunț necesită construirea în prealabil a unui șir conținând numărul total de persoane virusate pe fiecare zi. Aceasta se poate realiza apelând pentru fiecare zi o funcție care însumează cazurile de boală pe baza unui algoritm care însumează elementele unui tablou bidimensional.

```

Subalgoritm Suma( $a, m, n$ ):
    Sum  $\leftarrow 0$  { returnează suma elementelor unui tablou bidimensional }
    pentru  $i=1, n$  execută
        pentru  $j=1, n$  execută
            sum  $\leftarrow$  sum +  $a[i, j]$ 
        sfârșit pentru
    sfârșit pentru
    Suma  $\leftarrow$  sum
sfârșit subalgoritm

```

Determinarea maximului din șirul sumelor o realizăm cu subalgoritmul cunoscut:

```

Subalgoritm Maxim( $a, n$ ):
    { returnează valoarea celui mai mare element dintr-un șir de numere }
    max  $\leftarrow a[1]$ 
    pentru  $i=2, n$  execută
        dacă max <  $a[i]$  atunci
            max  $\leftarrow a[i]$ 
        sfârșit dacă
    sfârșit pentru
    maxim  $\leftarrow$  max
sfârșit subalgoritm

```

După ce numărul maxim de îmbolnăviri a fost determinat, nu rămâne decât să se afișeze zilele în care această valoare maximă a fost atinsă (vezi algoritmul principal).

3. Pentru a rezolva cel de-al *doilea subpunct*, trebuie să verificăm pentru fiecare individ dacă a fost continuu bolnav pe parcursul celor p zile. Această cerință este rezolvată în algoritmul principal, în care am notat cu s șirul sumelor de indivizi bolnavi, pe fiecare zi.

Algoritm Virus:

```

Prima_zi(a,m,n,p):
    { generarea tabloului tridimensional cuprinzând evoluția virusărilor }
Generează(a,m,n);
    { 1. se construiește șirul numărului total de indivizi virusați pentru fiecare zi }
pentru zi=1,p execută:
    s[zi] ← Suma(a[zi],m,n)
sfârșit pentru
    { se determină numărul maxim de indivizi virusați }
număr_maxim ← Maxim(S,p)
    { se afișează zilele în care s-a atins gradul maxim de virusare }
pentru zi=1,p execută:
    dacă s[zi] = număr_maxim atunci
        scrie zi
    sfârșit dacă
sfârșit pentru
    { 2. căutăm persoane bolnave numai dacă numărul p de zile este mai mic }
    { decât numărul maxim de zile în care virusul trăiește }
dacă p < 30 atunci
    pentru i=1,n execută:
    pentru j=1,n execută:
        Virusat ← adevărat { inițializare - marcăm individul ca fiind virusat }
    pentru zi=1,p execută:
        dacă a[zi,i,j] = 0 atunci
            Virusat ← fals { dacă cel puțin o zi nu a fost virusat, îl demarcăm }
        sfârșit dacă
        dacă Virusat atunci
            scrie i, j
        sfârșit dacă
        sfârșit pentru
        sfârșit pentru
        sfârșit pentru
        sfârșit dacă
sfârșit algoritm

```