

## 2.2.3 Probleme propuse

1. Se consideră un șir de  $n$  valori naturale de cel mult 9 cifre. Determinați numărul de valori care pot fi scrise sub forma  $k! (1*2*3*...*k)$ . Nu se vor folosi date structurate pentru reținerea celor  $n$  numere. În cadrul programului, se vor defini două subprograme recursive:

- funcția *Ok*, care verifică dacă o valoare primită printr-un parametru reprezintă factorialul unei valori. Funcția returnează, în Pascal, valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- funcția *Nr*, care permite citirea celor  $n$  numere și returnează numărul de valori factoriale. În cadrul ei se va apela funcția *Ok*.

*Exemplu:* Pentru  $n=5$  și valorile 7, 16, 6, 13, 24, se va afișa 2, deoarece  $6=1*2*3$ ,  $24=1*2*3*4$ .

2. Considerăm un tablou unidimensional  $a$  cu  $n(n<50)$  elemente numere naturale. Realizați un program care afișează, pe linii, elementele din vector grupate după cifra dominantă (prima în scrierea zecimală). Pe aceeași linie vor fi scrise elemente cu aceeași cifră dominantă. În cadrul programului, se vor defini două subprograme recursive:

- funcția *Cif*, care determină cifra dominantă a unei valori primite printr-un parametru.
- subprogramul *Afis*, care afișează pe o linie a ieșirii standard elementele din vector ce au cifra dominantă egală cu o valoare transmisă prin parametrul  $c$ . Parametrul întreg  $i$  va indica poziția(indicele) elementului curent din vector.

*Exemplu:* Pentru  $n=7$  și  $a=(334, 124, 21, 34, 122, 1, 39)$ , se va afișa:

124 122 1

21

334 34 39

3. Considerăm un tablou unidimensional  $a$  cu  $n(n<50)$  elemente numere naturale de cel mult 9 cifre. Să se realizeze un program care ordonează elementele vectorului, crescător după numărul de cifre pare pe care le conțin. În situația elementelor cu același număr de cifre pare, ordonarea se va face în funcție de valorile acestora. În cadrul programului, se vor defini două subprograme recursive:

- funcția *Nr*, care determină numărul de cifre pare al unei valori primite printr-un parametru.
- subprogramul *Sortare*, care efectuează ordonarea crescătoare a elementelor vectorului, după regula dată. Acesta va face apel la funcția *Nr* și nu va conține instrucțiuni repetitive.

*Exemplu:* Pentru  $n=7$  și  $a=(384, 824, 21, 34, 122, 1268, 39)$ , se va afișa:  
39, 21, 34, 122, 384, 824, 1268,

4. Considerăm un tablou unidimensional  $a$  cu  $n(n < 50)$  elemente numere naturale de cel mult 9 cifre. Să se realizeze un program care permite ștergerea elementului care are cel mai mare produs al cifrelor impare ce apar în scrierea sa. Dacă există mai multe elemente cu această proprietate, se va elimina cel de indice minim.

În cadrul programului, se vor defini două subprograme recursive:

- funcția  $P$ , care determină produsul cifrelor impare ale unei valori primite printr-un parametru.
- subprogramul  $Del$ , care permite ștergerea unui element dintr-un vector, al cărui indice este transmis prin parametrul întreg  $i$ . Subprogramul va avea doi parametri referință, reprezentând tabloul din care se efectuează ștergerea și lungimea acestuia. Ștergerea se va face prin deplasarea elementului  $a[i+1]$  pe poziția  $i$ , ș.a.m.d.

*Exemplu:* Pentru  $n=7$  și  $a=(384, 824, 21, 349, 122, 1268, 37)$ , se va afișa vectorul cu șase elemente:  $(384, 824, 21, 122, 1268, 37)$ .

5. Considerăm un tablou unidimensional  $a$  cu  $n(n < 50)$  elemente numere naturale de cel mult 9 cifre. Să se realizeze un program care determină cel mai mare divizor comun al elementelor care nu sunt numere prime. În cadrul programului, se vor defini două subprograme recursive:

- funcția  $Ok$ , care verifică dacă o valoare primită printr-un parametru reprezintă un număr prim. Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- funcția  $Cmmmd$ , care returnează cel mai mare divizor comun a două valori transmise prin intermediul unor parametri valoare.

*Exemplu:* Pentru  $n=7$  și  $a=(37, 40, 13, 60, 31, 11, 140)$ , se va afișa 20.

6. Considerăm un tablou unidimensional  $a$  cu  $n(n < 50)$  elemente numere naturale de cel mult 9 cifre. Să se realizeze un program care înlocuiește fiecare element cu valoarea lui din care s-au eliminat toate aparițiile cifrei maxime. Astfel, elementul 34241 va deveni 321. În cadrul programului, se vor defini trei subprograme recursive:

- funcția  $Cmax$ , care returnează cifra maximă a unei valori primite printr-un parametru întreg-lung.
- funcția  $DelCif$ , care primește prin parametrii  $x$  și  $c$ , două valori întregi. Aceasta returnează valoarea obținută din  $x$  prin ștergerea cifrei  $c$ .
- subprogramul  $S$ , care parcurge elementele vectorului primit printr-un parametru referință și le înlocuiește cu valorile ce respectă cerința din enunț. Subprogramul primește indicele elementului curent printr-un parametru valoare.

*Exemplu:* Pentru  $n=7$  și  $a=(37, 443, 13, 160, 31, 11, 140)$ , se va afișa  
 $a=(3, 3, 1, 10, 1, 0, 10)$ .

7. Considerăm un tablou unidimensional  $a$  cu  $n(n < 100)$  elemente numere naturale de cel mult 4 cifre. Să se realizeze un program care înlocuiește fiecare element cu cel mai apropiat palindrom de acesta. Astfel elementul 341 va deveni 343, iar 146 va deveni 141. În cadrul programului, se vor defini două subprograme recursive:

- funcția *Pal*, care verifică dacă o valoare primită prin parametru  $x$  reprezintă un număr palindrom. Parametrul referință  $z$  va memora, în final, inversul numărului  $x$ . Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- funcția *Search*, care primește prin parametrii  $x$  și  $t$ , două valori întregi. Aceasta returnează cel mare palindrom mai mic ca  $x$  sau cel mai mic palindrom mai mare ca  $x$ , după cum valoarea la apel a parametrului  $t$  este -1 sau 1. Subprogramul va face apel la funcția *Pal*.

*Exemplu:* Pentru  $n=4$  și  $a=(37, 44, 130, 16)$ , se va afișa  $a=(33, 44, 131, 11)$ .

8. Considerăm un tablou unidimensional  $a$  cu  $n(n < 100)$  elemente numere naturale de cel puțin două cifre. Să se realizeze un program care înlocuiește elementul cu număr maxim de apariții din vector cu suma primelor două cifre ale sale. În cadrul programului, se vor defini trei subprograme recursive:

- funcția *Sc*, care returnează suma primelor două cifre ale unei valori primite printr-un parametru întreg.
- funcția *Ap*, care primește, prin parametrii  $x$  și  $i$ , două valori întregi. Aceasta determină numărul de apariții al lui  $x$  în vectorul  $a$ . Valoarea transmisă prin parametrul  $i$  indică poziția curentă în vector.
- subprogramul *Max*, care primește, prin parametrul  $i$ , indicele elementului curent și returnează, prin parametrul referință  $x$ , elementul cu număr maxim de apariții în vectorul  $a$ . Subprogramul apelează funcția *Ap*.

*Exemplu:* Pentru  $n=7$  și  $a=(3700, \underline{544}, 130, \underline{544}, 3700, \underline{544}, 130)$ , se va afișa  $a=(3700, 9, 130, 9, 3700, 9, 130)$ .

9. Considerăm un tablou unidimensional  $a$  cu  $n(n < 100)$  elemente numere naturale de cel mult patru cifre. Să se realizeze un program care șterge toate aparițiile elementului minim. În cadrul programului, se vor defini trei subprograme recursive:

- funcția *Min*, care determină elementul minim al vectorului  $a$ . Parametrul  $i$  indică poziția curentă în vector.
- subprogramul *Del*, care permite ștergerea elementului din  $a$  al cărui indice este transmis prin parametrul  $i$ . Subprogramul va avea și doi parametri referință reprezentând tabloul din care se efectuează ștergerea și lungimea acestuia. Ștergerea se va face prin deplasarea elementului  $a[i+1]$  pe poziția  $i$ , ș.a.m.d.

*Exemplu:* Pentru  $n=7$  și  $a=(37, \underline{4}, 130, \underline{4}, 37, \underline{4}, 130)$ , se va afișa 37, 130, 37, 130.

10. Se consideră un vector ce conține  $n$  cifre zecimale ( $n < 10$ ). Să se verifice dacă numerele formate cu cifrele din vector citite de la dreapta la stânga și de la stânga la dreapta reprezintă cuburi perfecte.

În cadrul programului, se vor defini două subprograme recursive:

- funcția *Nr*, care determină numărul format cu elementele vectorului *a*. Parametrul *i* indică poziția curentă în vector, iar parametrul *t* sensul de parcurgere al tabloului. Dacă la apel  $t=-1$ , atunci numărul va fi format cu cifrele vectorului de stînga la dreapta și de la dreapta spre stînga, dacă  $t=1$ .
- funcția *Cub*, care primește două valori întregi prin intermediul parametrilor *x* și *i*. Aceasta verifică există o valoare *i* astfel încât *x* se poate scrie ca  $i^3$ . Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.

*Exemplu* :  $n=3$  și vectorul: (1, 2, 5), se va afișa „125 este cub perfect, 521 nu este un cub perfect”.

**11.** Se consideră o valoare *n* naturală, de cel mult 9 cifre. Să se descompună *n* în toate modurile posibile ca sumă de două numere oglindite.

În cadrul programului, se vor defini două subprograme recursive:

- funcția *Inv*, care determină inversul unui număr transmis parametrul valoare *x*. Parametrul referință *y* va memora inversul construit succesiv.
- subprogramul *S*, care primește două valori întregi prin intermediul parametrilor *x* și *i*. Acesta afișează toate valorile *i* cu proprietatea că suma dintre *i* și inversul său este egală cu *x*.

*Exemplu* : Pentru  $n=787$ , se va afișa: 146+641, 245+542, 344+443.

**12.** Considerăm un șir de *n* valori naturale ( $n \leq 50$ ) reținute în tabloul unidimensional *a*. Realizați un program care construiește alți doi vectori ce vor conține elementele lui *a* care sunt numere perfecte, respectiv pe cele care nu sunt numere perfecte. Un număr egal cu suma divizorilor săi strict mai mici decât el se numește număr perfect.

În cadrul programului, se vor defini două subprograme recursive:

- funcția *Sd*, care returnează numărul de divizori ai unei valori întregi transmise ca parametru.
- subprogram recursiv *Make*, care construiește alți doi vectori ce vor conține elementele lui *a* care sunt numere perfecte, respectiv cele care nu reprezintă valori perfecte. Subprogramul va returna vectorii construiți și lungimile acestora prin intermediul parametrilor. Acesta va apela funcția recursivă *Sd*.

*Exemplu* : Pentru  $n=5$  și  $a=(10, 6, 21, 28, 496)$ , programul va construi doi vectori de lungime 3 respectiv 2: (6, 28, 496) și (10, 21).

**13.** Se citește, cu ajutorul unui șir de caractere, o expresie ai căror operanzi se consideră a fi doar parantezele rotunde. Astfel, o valoare inclusă între paranteze rotunde va fi convertită în baza 2. Să se afișeze valorile obținute după conversia în baza 2 a tuturor operanzilor prezenți în expresie.

În cadrul programului, se vor defini două subprograme recursive:

- subprogramul *B2*, care afișează cifră cu cifră valoarea obținută în urma conversiei în baza 2 a unui număr întreg transmis ca parametru.

- subprogramul *Extrag*, care primește, prin intermediul parametrului *s*, un șir de caractere reprezentând o expresie descrisă anterior. Subprogramul afișează valorile operanzilor obținute în urma conversiei în baza 2. Se va face apel la funcția *B2*.

*Exemplu* : Pentru expresia (7)(8)(13), se va afișa 111, 1000, 1101.

14. Se citesc de la tastatură *n* cuvinte formate din cel mult 50 de litere ale alfabetului englez și o secvență *s* de caractere considerată o silabă. Realizați un program care afișează cuvintele care conțin cele mai multe silabe egale cu *s*. Nu se vor folosi date structurate pentru reținerea celor *n* cuvinte. În cadrul programului, vor fi definite următoarele două subprograme recursive:

- funcția *Nr*, care primește două șiruri de caractere prin parametrii *x* și *y*. Aceasta determină numărul de apariții a lui *x* în *y*.
- subprogramul *Solve*, care permite citirea celor *n* cuvinte și afișarea celor cu proprietatea specificată în enunț. Subprogramul va avea trei parametri: parametrul valoare *i*, care indică numărul de ordine al cuvântului care va fi citit, parametrul valoare *s*, care reprezintă silaba căutată și parametrul referință *max*, ce va memora numărul maxim de apariții al lui *s* printre cuvintele citite.

*Exemplu* : Pentru *n=4*, silaba *ma* și cuvintele *mama*, *tamara*, *mamaie*, *inca*, se va afișa *mamaie*, *mama*

15. Se citesc de la tastatură *n* cuvinte formate din cel mult 50 de litere ale alfabetului englez. Să se realizeze un program care afișează, în ordinea inversă citirii, cuvintele oglindite, din care lipsesc vocalele. Nu se vor folosi date structurate pentru reținerea celor *n* cuvinte. În cadrul programului, vor fi definite următoarele două subprograme recursive:

- funcția *Inv*, care primește un șir de caractere prin parametrul *x* și returnează inversul șirului din care lipsesc vocalele.
- subprogramul *Solve*, care permite citirea celor *n* cuvinte și afișarea în ordine inversă a valorilor cerute prin enunț. Subprogramul va avea un parametru valoare *i*, care indică numărul de ordine al cuvântului care va fi citit. Acesta va apela funcția *Inv*.

*Exemplu* : Pentru *n=4* și cuvintele *mama*, *tamara*, *mamaie*, *inca*, se va afișa *cn*, *mm*, *rmt*, *mm*.

16. Se citesc de la tastatură *n* numere de cel mult patru cifre. Să se realizeze un program care determină produsul numerelor care sunt egale cu suma factorialelor cifrelor componente. Nu se vor folosi date structurate pentru reținerea celor *n* numere. În cadrul programului, vor fi definite următoarele trei subprograme recursive:

- funcția *Fact*, care primește o cifră prin parametrul *x* și returnează valoarea *x!*
- funcția *Sum*, care primește un număr întreg prin parametrul *x* și returnează suma factorialelor cifrelor lui *x*. Aceasta va face apel la funcția *Fact*.

- funcția *Prod*, care permite citirea celor  $n$  valori întregi și care returnează produsul valorilor cerute.

*Exemplu:* Pentru  $n=5$  și numerele 1, 63, 2, 145, 496, se va afișa 290, deoarece  $290 = 1 \cdot 2 \cdot 145$ .

17. Considerăm șirul 0, 2, 1, 3, 2, 4, 3, 5, 4, 6....Să se realizeze cu program care permite memorarea primilor  $n^2$  termeni ai șirului, într-o matrice pătratică cu  $n$  linii și  $n$  coloane ( $n < 10$ ). Termenii vor fi plasați în ordine pe linii de la stânga la dreapta. În cadrul programului, se vor defini următoarele două subprograme recursive:

- funcția recursivă *T*, care returnează termenul șirului cu numărul de ordine transmis prin intermediul unui parametru întreg.
- subprogramul recursiv *M*, care permite memorarea primilor  $n^2$  termeni ai șirului, într-o matrice pătratică cu  $n$  linii și  $n$  coloane ( $n < 10$ ). Subprogramul *M* primește două valori naturale prin parametrii  $i$  și  $j$ , valori care reprezintă indicii liniei și coloanei elementului curent. Matricea creată este returnată prin parametrul referință  $a$ .

*Exemplu:* Pentru  $n=4$ , programul va afișa matricea:

0 2 1 3

2 4 3 5

4 6 5 7

6 8 7 9

18. Fie un tablou bidimensional  $A(n,m)$ . Realizați un program care inversează elementele de pe liniile care încep cu un număr prim. În cadrul programului, vor fi definite următoarele subprograme recursive:

- funcția *Ok*, care verifică dacă o valoare primită printr-un parametru reprezintă un număr prim. Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- subprogramul *Inv*, care primește doi parametri întregi  $i$  și  $l$ . Acesta inversează elementele de pe linia  $l$  prin interschimbarea elementul  $a[l,i]$  cu cel simetric în cadrul liniei.
- subprogramul *Pl*, care parcurge fiecare linie a tabloului și inversează liniile care încep cu un număr prim. El va primi, printr-un parametru valoare, indicele liniei curente.

*Exemplu:* Pentru  $n=3$ ,  $m=4$  și matricea:

4 2 3 7

2 3 4 5

3 4 5 2

se va afișa :

4 2 3 7

5 4 3 2

2 5 4 3

19. Se consideră un tablou bidimensional cu  $n$  linii și  $n$  coloane ( $1 \leq n \leq 100$ ), având componente de tip întreg. Cele două diagonale ale tabloului împart tabloul în patru regiuni în formă de triunghi. Se cere să se determine suma componentelor din interiorul fiecărei zone. În cadrul programului, vor fi definite patru funcții

recursive. Fiecare dintre ele calculează suma elementelor situate într-una dintre cele patru zone. Toate funcțiile vor primi, prin doi parametri valoare, indicii liniei și elementului curent. În cadrul nici unei funcții nu se vor folosi instrucțiuni repetitive.

*Exemplu:* Pentru  $n=5$  și tabloul:

0 1 1 1 0  
2 0 1 0 3  
2 2 0 3 3  
2 0 4 0 3  
0 4 4 4 0

se va afișa:

S1=4

S2=8

S3=12

S4=16

**20.** Să se rearanjeze elementele unei matrice de dimensiune  $n \times m$ , astfel încât ele să fie ordonate crescător atât pe linii cât și pe coloane. În cadrul programului, vor fi definite următoarele subprograme recursive:

- Subprogramul *OrdL*, care ordonează elementele situate pe o linie al cărui indice este transmis printr-un parametru valoare. Subprogramul nu conține nici o structură repetitivă.
- Subprogramul *OrdC*, care ordonează elementele situate pe o coloană al cărui indice este transmis printr-un parametru valoare. Subprogramul nu conține nici o structură repetitivă.

*Exemplu:*  $n=3$ ,  $m=4$  și matricea

3 1 8 9  
4 6 5 7  
2 0 1 3

se va afișa:

0 1 2 3

1 3 6 7

4 5 8 9

**21.** Se consideră tabloul bidimensional  $a$  cu  $n$  linii și  $n$  coloane ( $1 \leq n \leq 100$ ), având componente de tip întreg. Realizați un program care înlocuiește fiecare element situat sub diagonală secundară cu suma exponenților ce apar la descompunerea lui în factori primi. De exemplu valoarea  $360=2^3 \cdot 3^2 \cdot 5^1$  va fi înlocuită cu numărul  $6(3+2+1)$ . În cadrul programului, se vor defini următoarele subprograme recursive:

- funcția *S\_e*, care returnează suma exponenților din descompunerea în factori primi a unui număr primit prin parametrul întreg  $x$ .
- subprogramul *Diag*, care parcurge toate elementele situate sub diagonală principală a lui  $a$ . Acestea vor fi înlocuite cu valori cerute în enunț. Subprogramul va avea doi parametri valoare, reprezentând indicele liniei și al coloanei elementului curent. Nu se vor folosi instrucțiuni repetitive.

*Exemplu:*  $n=4$  și matricea

123 211 213 901  
124 216 215 360  
122 125 125 18  
535 400 625 120

se va afișa:

123 211 213 901

124 216 215 6

122 125 3 3

535 6 4 5

22. Se citește de la tastatură o valoare naturală de cel mult 5 cifre. Să se realizeze un program care permite memorarea primelor  $n^2$  numere prime mai mici decât  $k$ , într-o matrice pătratică cu  $n$  linii și  $n$  coloane. Programul va determina cea mai mare valoare posibilă pentru  $n$ . Termenii vor fi plasați în ordine pe linii de la stânga la dreapta. În cadrul programului, se vor defini următoarele două subprograme recursive:

- funcția  $T$ , care verifică dacă o valoare naturală transmisă printr-un parametru reprezintă un număr prim. Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- subprogram  $M$ , care permite memorarea primelor  $n^2$  numere prime într-o matrice pătratică cu  $n$  linii și  $n$  coloane. Subprogramul primește trei valori naturale prin parametrii  $i$ ,  $j$  și  $v$ , valori care reprezintă indicii liniei și coloanei elementului care va memora numărul  $v$ , doar dacă acesta este prim. Se va face apel la funcția *Ok*.

*Exemplu:* Pentru  $k=30$ , programul va determina  $n=3$  și va genera matricea:

```
2 3 5
7 11 13
17 19 23
```

23. Realizați un program care generează toate cuvintele de lungime  $n$  formate din cele două caractere ale codului Morse '.', '-.'. În cadrul programului, se va defini subprogramul recursiv *Morse* care generează și afișează, pe câte o linie a ieșirii standard, șirurile de caractere cerute. Subprogramul va avea un singur parametru valoare de tip șir de caractere.

*Exemplu:* Pentru  $n=2$  se va afișa: .. .- -. --

24. Se consideră un tablou unidimensional de lungime  $n$ . Asupra acestuia se vor efectua operații de "tăiere" care constau în "înjumătățirea" lui și îndepărtarea unuia dintre cele două subtablouri astfel obținute. Dacă  $n$  este impar, elementul de pe poziția  $[n/2]$  este eliminat. Procesul de tăiere se repetă până la obținerea unui tablou cu un singur element. Să se afișeze, pe o singură linie în fișierul text *Out.txt*, toate elementele care se pot obține la încheierea procesului de tăiere.

Datele de intrare se citesc din fișierul *In.txt* în următorul format: Pe prima linie numărul  $n$ , iar pe următoarea linie elementele separate prin câte un spațiu.

<i>Exemplu:</i> Pentru fișierul <i>In.txt</i>	<i>Out.txt</i>
7	1 3 5 7
1 2 3 4 5 6 7	