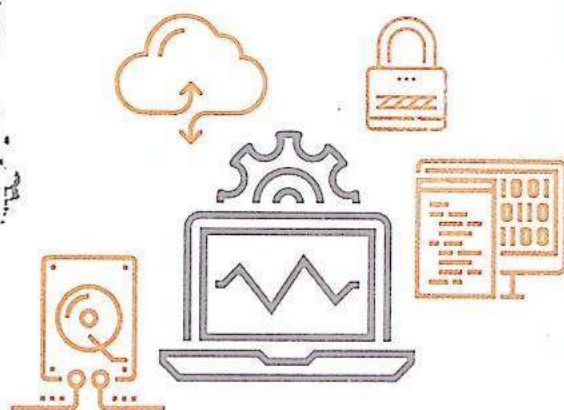




Avizat
M.E.

Silvia Grecu
Lucia Miron
Mirela Țibu

Bacalaureat



INFORMATICĂ

Limbajul C++

- _ Ghid complet de pregătire a examenului de Bacalaureat
- _ 25 de teste – după modelul M.E. – pentru specializarea Matematică-Informatică
- _ 10 teste – după modelul M.E. – pentru specializarea Științe ale Naturii

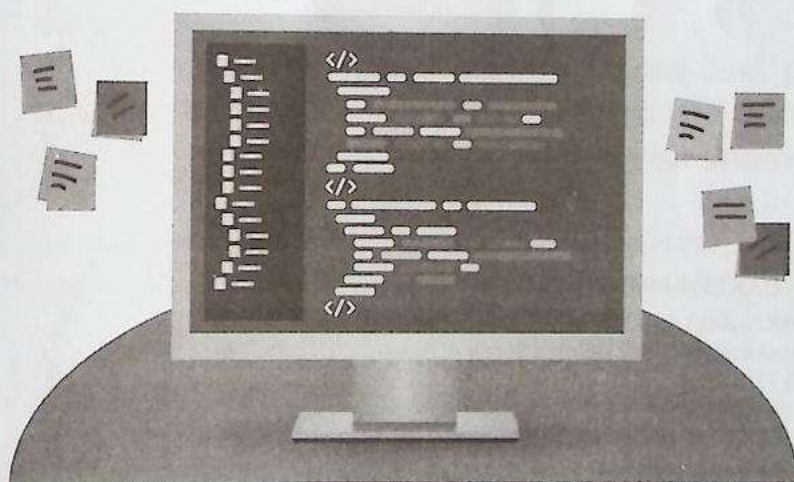


EDITURA PARALELA 45
EDUCAȚIONAL

SILVIA GRECU
LUCIA MIRON
MIRELA ȚIBU

BACALAUREAT INFORMATICĂ LIMBAJUL C++

Ghid complet de pregătire a examenului de Bacalaureat



Specializările:



```
{ Matematică-Informatică  
  Științe ale Naturii  
}
```

Editura Paralela 45

Lucrarea este realizată în conformitate cu programa examenului național de Bacalaureat la disciplina Informatică.

Redactare: Iuliana Voicu
Tehnoredactare: Mariana Dumitru
Pregătire de tipar: Marius Badea
Design copertă: Mirona Pintilie

Descrierea CIP a Bibliotecii Naționale a României

GRECU, SILVIA

Bacalaureat - Informatică : limbajul C++ : ghid complet de pregătire
a examenului de Bacalaureat : specializările Matematică-Informatică,
Științe ale Naturii / Silvia Grecu, Lucia Miron, Mirela Țibu. - Pitești :
Paralela 45, 2019

Conține bibliografie

ISBN 978-973-47-3115-2

I. Miron, Lucia

II. Țibu, Mirela-Anca

004

COMENZI - CARTEA PRIN POȘTĂ

EDITURA PARALELA 45

Bulevardul Republicii, nr. 148, Clădirea C1, etaj 4, Pitești,
jud. Argeș, cod 110177

tel.: 0248 633 130; 0753 040 444; 0721 247 918

tel./fax: 0248 214 533; 0248 631 439; 0248 631 492

e-mail: comenzi@edituraparelela45.ro

www.edituraparelela45.ro

Tiparul executat la tipografia Editurii Paralela 45

E-mail: tipografie@edituraparelela45.ro

Copyright © Editura Paralela 45, 2019

Prezenta lucrare folosește denumiri ce constituie mărci înregistrate,
iar conținutul este protejat de legislația privind dreptul de proprietate intelectuală.

Cuvânt-înainte

Lucrarea de față se dorește a fi un ghid de pregătire individuală a elevilor de liceu pentru proba de Informatică a examenului de Bacalaureat – varianta C++. În acest sens, cele unsprezece capitole de teorie prezintă un breviar al noțiunilor studiate la disciplina Informatică, prezente în programa de Bacalaureat, urmat de modele de teste propuse prin a căror rezolvare se poate realiza aprofundarea acestor noțiuni.

După parcurgerea noțiunilor de teorie, cartea se continuă cu teze întocmite după modelul examenului de Bacalaureat, atât pentru specializarea Matematică-Informatică, cât și pentru specializarea Științe ale Naturii.

Pentru a veni în sprijinul celor care vor folosi această carte, problemele și tezele au fost rezolvate integral, iar programele au fost verificate în Code::Blocks.

Ghidul poate fi utilizat în aceeași măsură și ca auxiliar didactic la orele de Informatică.

Sperăm că acest ghid se va dovedi un instrument util elevilor, dar și profesorilor care îl folosesc în pregătirea examenului de Bacalaureat.

Autoarele

PARTEA I

BREVIAR



CAPITOLUL 1

Reprezentarea algoritmilor în pseudocod. Elemente de bază ale limbajului C++

Teorie

Principiul programării structurate: Orice algoritm poate fi descris utilizând trei tipuri de structuri de control: liniară, alternativă și repetitivă.

Structuri de control – reprezentare în pseudocod	Implementare C++
<p>• Structura liniară – conține operații de citire/scriere/atribuire.</p> <p>citește $var_1, var_2, \dots, var_n$</p> <p>$var_1, var_2, \dots$ sunt identificatori de variabile</p> <p>scrie $expresie_1, expresie_2, \dots, expresie_n$</p> <p>$expresie_1, expresie_2, var_2, \dots$ sunt expresii de orice tip (numeric, caracter sau șir de caractere)</p> <p>$variabila \leftarrow expresie$</p> <p>Se evaluează expresia din partea dreaptă, iar valoarea obținută i se atribuie variabilei cu identificatorul <i>variabila</i>.</p>	<p>Citirea valorilor variabilelor de la tastatură:</p> <p><code>cin >> var₁ >> var₂ >> ... >> var_n;</code></p> <p>//valorile variabilelor se citesc în ordinea din listă</p> <p>Afișarea valorilor expresiilor pe ecran:</p> <p><code>cout << expresie₁ << ... << expresie_n;</code></p> <p>//expresiile se vor evalua și afișa în ordinea din listă</p> <p><code>variabila = expresie;</code></p> <p><code>variabila < op > = expresie;</code></p> <p>unde $op \in \{+, -, *, /, \%, \gg, \ll, \&, , ^\}$, este echivalentă cu</p> <p><code>variabila = variabila <op> expresie;</code></p>
<p>Structura alternativă – conține operații de decizie:</p> <p>dacă $\langle expresie \rangle$ atunci secvența_A altfel secvența_B</p>	<p>Instrucțiunile <i>if</i> și <i>switch</i> implementează structura alternativă.</p> <p><code>if (expresie)</code> instrucțiuni_A <code>else</code> instrucțiuni_B</p> <p>Efect: Se evaluează <i>expresie</i>. Dacă valoarea este diferită de 0, se execută secvența de</p>

Sau

[dacă <expresie> atunci
 secvența_A

Efect: Se evaluează *expresie*. Dacă este adevărată, se execută *secvența_A*, altfel se execută *secvența_B* și se continuă algoritmul cu următoarele structuri.

Secvențele A și/sau B pot conține orice alte structuri liniare, alternative sau repetitive.

Ramura altfel poate lipsi.

instrucțiuni *A*, altfel se execută secvența de instrucțiuni *B*.

În cazul în care o secvență conține mai mult de o instrucțiune, se va delimita prin { } grupul de instrucțiuni care definesc secvența.

Instrucțiunea *switch* pentru alternative multiple, simplifică modul de scriere pentru if-urile imbricate.

```
switch (expresie) {
    case constantă_1: instrucțiuni_1
                        break;
    case constantă_2: instrucțiuni_2
                        break;
    .....
    case constantă_n: instrucțiuni_n
                        break;
    default: instrucțiuni
}
```

Efect: Valoarea *expresie* este evaluată la un tip întreg, apoi această valoare este comparată cu fiecare constantă. Este rulat blocul de instrucțiuni al valorii găsite.

Dacă este prezentă instrucțiunea *break* după blocul executat, se va părăsi instrucțiunea *switch*, altfel se vor executa în continuare toate blocurile de instrucțiuni ale constantelor care urmează, până la întâlnirea primului *break* sau până la sfârșitul instrucțiunii *switch*. În caz că numărul nu este egal cu niciuna dintre constante, este executat blocul aflat după *default*.

Structura repetitivă

• Cu condiție inițială

[cât timp <expresie> execută
 secvența_A

Efect:

Pas 1. Se evaluează *expresie*.

Pas 2. Dacă este diferită de 0, se execută *secvența_A* și se reia Pas 1. Dacă este 0 se părăsește structura repetitivă și se continuă algoritmul cu următoarele structuri.

Instrucțiunea *while* implementează structura repetitivă cu condiție inițială.

```
while (expresie)
{
    //declarări de date locale
    Instrucțiuni
}
```

Variabilele declarate în blocul instrucțiunii *while* sunt vizibile doar în acest bloc și sunt valabile doar pe durata execuției instrucțiunii.

Obs. 1) Dacă la prima evaluare, valoarea *expresiei* este 0, *secvența_A* nu se execută deloc.

2) În *secvența_A* trebuie să se modifice valoarea cel puțin a unei variabile care apare în *expresie* astfel încât, după un număr finit de pași, execuția structurii *cât timp* să se încheie.

- **Cu condiție finală**

```
repetă  
    secvența_A  
■ până când <expresie>
```

Efect:

Pas 1. Se execută *secvența_A*.

Pas 2. Se evaluează *expresie*.

Pas 3. Dacă valoarea este diferită de 0 (este adevărată), se părăsește structura repetitivă și se continuă algoritmul cu următoarele structuri. Dacă valoarea este egală cu 0, se reia Pasul 1.

Obs. 1) *secvența_A* se execută cel puțin o dată, indiferent de valoarea inițială a *expresiei*.

2) În *secvența_A* trebuie să se modifice valoarea cel puțin a unei variabile care apare în *expresie* astfel încât, după un număr finit de pași, execuția structurii *cât timp* să se încheie.

- **Cu număr cunoscut de pași**

```
pentru contor ← start, stop, pas execută  
    secvența_A  
■
```

Obs. Dacă *pas* nu este precizat, implicit va avea valoarea 1.

Efect:

Pas 1. Se atribuie variabilei *contor* valoarea de start (evaluată ca număr întreg)

Pas 2. Se compară valoarea *contor* cu valoarea *stop*, în funcție de semnul pasului. Dacă pasul este 1 (parcursere crescătoare) atunci compararea va fi *contor ≤ stop*, iar dacă pasul este -1 (parcursere descrescătoare), atunci compararea va fi *contor ≥ stop*

Instrucțiuni pot fi orice instrucțiuni C++, inclusiv dintre cele care implementează structuri repetitive.

Instrucțiunea **do-while** implementează structura repetitivă cu condiție finală.

```
do {  
    //declarări de date locale  
    Instrucțiuni  
} while (not expresie);
```

Atenție! Condiția de oprire a buclei, exprimată de *expresie* din descrierea în pseudocod, se transformă în condiție de repetare a buclei, în instrucțiunea din C++. De aceea apare negația (not *expresie*).

Variabilele declarate în blocul instrucțiunii *do-while* sunt vizibile doar în acest bloc și sunt valabile doar pe durata execuției instrucțiunii.

Instrucțiuni pot fi orice instrucțiuni C++, inclusiv dintre cele care implementează structuri repetitive.

Instrucțiunea *for* implementează structura repetitivă cu număr cunoscut de pași *pentru*.

Dacă *pas* este 1

```
for (contor=start; contor<=stop;  
    contor++)  
{  
    Instrucțiuni  
}
```

Pas 3. Dacă este adevărată evaluarea expresiei corespunzătoare $contor \leq stop$ (pentru $pas = 1$) sau $contor \geq stop$ (pentru $pas = -1$) se execută *secvența A* și se trece la Pas 4.

Dacă este falsă, se părăsește structura repetitivă *pentru* și se continuă algoritmul cu următoarele structuri.

Pas 4. La valoarea *contor* se adună valoarea *pas* și se reia Pas 2.

Dacă *pas* este -1

```
for (contor = start; contor >= stop;
    contor --)
{
    Instrucțiuni
}
```

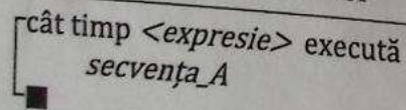
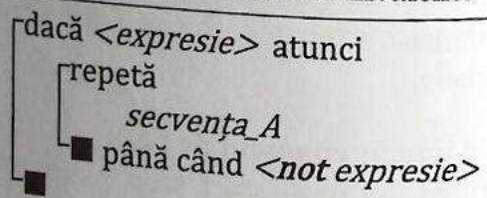
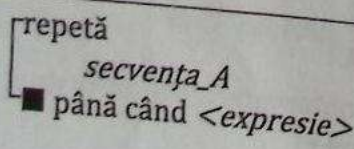
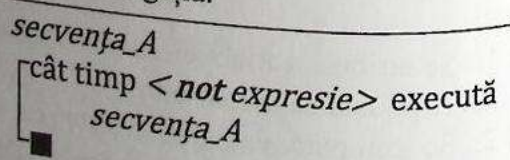
Sintaxa C++ a instrucțiunii **for** este

```
for (expresie1; expresie2; expresie3)
{
    Instrucțiuni
}
```

și este echivalentă cu

```
expresie1
while (expresie2)
{
    Instrucțiuni
    expresie3
}
```

Echivalența structurilor de control repetitive

Structura de control	Structuri de control echivalente
	 <p>Atenție! Condiția de repetare exprimată prin <i>expresie</i> în structura <i>cât timp</i> devine condiție de repetare în structura <i>repeta-până când</i>, de aceea apare negația.</p>
	 <p>Atenție! Condiția de oprire exprimată prin <i>expresie</i> în structura <i>repeta-până când</i> devine condiție de repetare în structura <i>cât timp</i>, de aceea apare negația.</p>

<pre> ┌ pentru <i>contor</i> ← start, stop execută │ <i>secvența_A</i> └ ──</pre>	<pre> <i>contor</i> ← start ┌ cât timp <i>contor</i> ≤ stop execută │ <i>secvența_A</i> │ <i>contor</i> ← <i>contor</i> + 1 └ ──</pre>
<pre> ┌ pentru <i>contor</i> ← start, stop execută │ <i>secvența_A</i> └ ──</pre>	<pre> <i>contor</i> ← start ┌ dacă <i>contor</i> ≤ stop atunci │ repetă │ <i>secvența_A</i> │ <i>contor</i> ← <i>contor</i> + 1 │ ── până când <i>contor</i> > stop └ ──</pre>
<pre> ┌ pentru <i>contor</i> ← start, stop, -1 │ execută │ <i>secvența_A</i> └ ──</pre> <pre> ┌ pentru <i>contor</i> ← start, stop, -1 │ execută │ <i>secvența_A</i> └ ──</pre>	<pre> <i>contor</i> ← start ┌ cât timp <i>contor</i> ≥ stop execută │ <i>secvența_A</i> │ <i>contor</i> ← <i>contor</i> - 1 └ ──</pre> <pre> <i>contor</i> ← start ┌ dacă <i>contor</i> ≥ stop atunci │ repetă │ <i>secvența_A</i> │ <i>contor</i> ← <i>contor</i> - 1 │ ── până când <i>contor</i> < stop └ ──</pre>
<pre> cât timp <i>val</i> > 0 execută ┌ <i>secvența_A</i> │ <i>val</i> ← <i>val</i> - 1 └ ──</pre>	<pre> ┌ pentru <i>val</i> ← val, 1, -1 execută │ <i>secvența_A</i> └ ──</pre> <p>sau</p> <pre> ┌ pentru <i>contor</i> ← val, 1, -1 execută │ <i>secvența_A</i> └ ──</pre> <p>Atenție la a doua implementare! În cazul în care <i>secvența_A</i> conține instrucțiuni care utilizează valoarea <i>val</i>, trebuie înlocuite toate referirile la <i>val</i> din <i>secvența_A</i> cu <i>contor</i>.</p>

Elemente de bază ale limbajului C/C++

Datele prelucrate de un program sunt *constante* și *variabile*. Unei variabile *i* se poate modifica valoarea pe parcursul execuției blocului în care este declarată, în timp ce unei constante nu i se poate schimba valoarea.

O *variabilă* este caracterizată prin nume (identificator de variabilă), tip, spațiu de memorie alocat și valoare. Spațiul de memorie alocat (numărul de octeți) și domeniul valorilor posibile depind de tipul variabilei.

Identificatorul unei variabile poate fi format din litere 'a'...'z', 'A'...'Z', cifre '0'...'9' și simbolul '_' și nu poate începe cu o cifră.

Corect: _3ab, max2, M_aux, p123

Inc corect: 1aB, 5_sum

În limbajul C/C++ pot fi reprezentate tipuri de date **simple** și **structurate** (omogene și neomogene).

Tipuri de date simple

Tip variabilă	Nr. octeți (bytes)	Domeniu de valori	Operatori	Funcții specifice
short int	2	$[-2^{15}, 2^{15} - 1]$	Operatori:	$\langle \text{cmath.h} \rangle$
unsigned short int	2	$[0, 2^{16} - 1]$	Aritmetici:	$\text{sqrt}(x), x \geq 0,$
int = long int	4	$[-2^{31}, 2^{31} - 1]$	$+, -, *, /, \%$	pentru $\sqrt{x}$
unsigned int	4	$[0, 2^{32} - 1]$	Pe biți:	$\text{abs}(x)$ pentru $ x $
= unsigned long int	8	$[0, 2^{32} - 1]$	$\gg, \ll, \&, , ^, \sim$	$\text{pow}(a,b)$ pentru a^b
long int	8	$[-2^{63}, 2^{63} - 1]$	Relaționali:	
long long int		$[-2^{63}, 2^{63} - 1]$	$<, <=, >=, >$	
unsigned long long		$[0, 2^{64} - 1]$	De egalitate:	
			$==, !=$	
float	4	$[-3.2 \cdot 10^{38}, 3.2 \cdot 10^{38}]$	Operatori:	$\langle \text{cmath.h} \rangle$
double	8	$[-1.7 \cdot 10^{308}, 1.7 \cdot 10^{308}]$	Aritmetici:	$\text{sqrt}(x), x \geq 0,$
			$+, -, *, /$	pentru $\sqrt{x}$
			Relaționali:	$\text{abs}(x)$ pentru $ x $
			$<, <=, >=, >$	$\text{pow}(a,b)$ pentru a^b
			De egalitate:	
			$==, !=$	
char	1	$[-128, 127]$	Operatori:	$\langle \text{ctype.h} \rangle$
unsigned char	1	$[0, 255]$	Aritmetici:	$\text{islower}(\text{car})$
char			$+, -, *, /, \%$	$= \begin{cases} 1, & \text{car e literă mică} \\ 0, & \text{altfel} \end{cases}$
			Relaționali:	$\text{isupper}(\text{car})$
			$<, <=, >=, >$	$= \begin{cases} 1, & \text{car e majusculă} \\ 0, & \text{altfel} \end{cases}$
			De egalitate:	$\text{tolower}(\text{car}) = \text{car}$
			$==, !=$	
				în minusculă = $\text{car} + 'a' - 'A'$
				$\text{toupper}(\text{car}) = \text{car}$
				în majusculă = $\text{car} - 'a' + 'A'$



limbajul C/C++ permite realizarea de conversii implicite sau explicite de tip a variabilelor și constantelor.

Situații care determină efectuarea implicită a unei conversii într-o expresie:

- dacă operanzii sunt de același tip real sau întreg, dar se memorează în locații cu dimensiuni diferite, conversia se efectuează spre locația cu dimensiune maximă;
- dacă operanzii sunt de tip diferit, întreg și real, conversia se va aplica numerelor întregi care devin numere reale;
- dacă un operand de tip **char** intervine într-o expresie aritmetică se va realiza conversia către tipul întreg, iar valoarea cu care se va evalua expresia va fi codul ASCII al caracterului;
- dacă valoarea reală a unei variabile sau a unei expresii este atribuită unei variabile întregi, se va realiza o conversie implicită spre tipul întreg care va determina „trunchierea” părții zecimale a numărului real.

Exemplul 1

.....

int a=12, b, ok ;

long long int n;

char ch1='A', ch2;

ch2=ch1+3;

//codul ASCII('A')=65, 65+3=68 si ch2='D', codul ASCII('D')=68

b=a%7*1.5;

//se efectuează $12\%7=5$, apoi $5*1.5=7.5$, care va fi trunchiat la partea întreagă 7, apoi atribuit lui b

ok=(a<b);

//se vor compara cele două valori ale variabilelor a și b ($12<7$) și se va atribui lui ok valoarea 0

cout <<ch2<<' '<<b<<' '<<ok; //se vor afișa: D 7 0

Exemplul 2

.....

float x, y;

int a=7, b=2;

x=a/2+5.0;

//se efectuează $7/2=3$ (operanzi întregi) apoi $3+5.0=8.0$ care se va atribui

lui x

y=(float)a/2+3.0; //se efectuează $7.0/2=3.5$ apoi $3.5+3.0=6.5$ care se va atribui

lui y

cout <<x<<' '<<y; //se vor afișa: 8 6.5

.....

Categorie	Operatori	Semnificație	Asociativitate
1. Prioritate unari de prioritate maximă	<code>()</code> , <code>[]</code> , <code>→</code> , <code>.</code>	Apel de funcție Expresie cu indici Selectorii de membru la structuri	St-Dr
2. Operatori unari	<code>!</code> <code>~, +, -</code> <code>++, --</code> <code>&, *</code> <code>sizeof(tip)</code> <code>(cast)</code>	Negare logică Negare bit cu bit (complementare cu 1) Plus și minus unari Incrementare/decrementare (pre și post) Obținerea adresei/indirectare Dimensiune operand (în octeți) Conversie explicită de tip – cast	Dr-St
3. Operatori aritmetici multiplicativi	<code>*</code> , <code>/</code> , <code>%</code>	Înmulțire/împărțire/restul împărțirii întregi	St-Dr
4. Adunare, scădere	<code>+</code> , <code>-</code>	Plus și minus binari	St-Dr
5. Deplasări	<code><<</code> , <code>>></code>	Deplasare stânga/dreapta pe biți	St-Dr
6. Relaționali	<code><</code> , <code><=</code> , <code>></code> , <code>>=</code>	Mai mic/ Mai mic sau egal/ Mai mare/ Mai mare sau egal	St-Dr
7. Egalitate	<code>==</code> , <code>!=</code>	Egal/ Diferit	St-Dr
8-10. Operatori logici pe biți	<code>&</code> , <code>^</code> , <code> </code>	ȘI/ SAU EXCLUSIV/ SAU logic bit cu bit	St-Dr
11-12. Operatori logici	<code>&&</code> , <code> </code>	ȘI/ SAU logic	St-Dr
13. Operator de atribuire compus	<code>=</code> <code><op>=</code>	Atribuire simplă: <i>variabila = expresie</i> ; Atribuire multiplă <i>variabila1 = variabila2 = ... = expresie</i> ; Atribuire compusa: <i>variabila <op>= expresie</i> unde <i>op</i> ∈ { <code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> , <code>%</code> , <code>>></code> , <code><<</code> , <code>&</code> , <code> </code> , <code>^</code> }, este echivalentă cu <i>variabila = variabila <op> (expresie)</i> ;	Dr-St
14. Operator condițional	<code>?:</code>	<i>expresieTest? variantaDa : variantaNu</i> ; Se evaluează <i>expresieTest</i> . Dacă este adevărată (diferită de 0), se evaluează doar <i>expresieDa</i> , altfel, se evaluează doar <i>expresieNu</i> .	Dr-St
15. Virgula	<code>,</code>	<i>expresie1, expresie2, ..., expresieN</i> ; Se evaluează expresiile în ordine iar valoarea finală este cea a <i>expresieN</i>	St-Dr

Expresiile sunt succesiuni de operatori și operanzi, corecte din punct de vedere sintactic. Orice expresie are o valoare, obținută la evaluarea acesteia.

În funcție de tipul operanzilor și al operatorilor, expresiile pot fi:

- aritmetice: valoarea obținută la evaluare este întreagă sau reală.
- logice: valoarea obținută la evaluare este 0 (fals) sau 1 (adevărat).

La evaluarea expresiilor logice se utilizează tabelele de valori ale operatorilor logici.

! (not), && (and), || (or)

!	0	1	&&	0	1		0	1
	1	0	0	0	0	1	0	1
			1	0	1	0	1	1

În cazul expresiilor logice compuse se pot aplica regulile lui de Morgan:

$$\neg(E1 \ \&\& \ E2) = \neg(E1) \ || \ \neg(E2)$$

$$\neg(E1 \ || \ E2) = \neg(E1) \ \&\& \ \neg(E2)$$

unde E1 și E2 sunt expresii logice sau aritmetice.

Structura unui program C++

```
#include <iostream.h>
//includerea altor librării necesare în program
using namespace std;
//declarare variabile globale
//declarare funcții utilizator
int main()
{
    //declarare variabile locale
    instrucțiuni
    return 0;
}
```

PROBLEME PROPUSE

1. Variabilele **a** și **b** reprezintă numere întregi nenule. Care dintre expresiile C/C++ au valoarea **1** dacă și numai dacă **a** și **b** au același semn și sunt impare?
 - 1) $a+b>0 \ \&\& \ a\%2!=0 \ \&\& \ b\%2!=0$
 - 2) $!(a\%2==0 \ || \ b\%2==0) \ \&\& \ !(a*b<0)$
 - 3) $a\%2+b\%2==2 \ \&\& \ a*b>0$
 - 4) $(a\%2>0 \ \&\& \ b\%2>0) \ || \ !(a\%2+b\%2<2)$
2. Variabila **n** reprezintă un număr întreg nenul, iar variabilele **a** și **b** reprezintă numere reale. Care dintre următoarele expresii C/C++ au valoarea **1** dacă și numai dacă **n** este multiplu de **2019** și **nu** aparține intervalului (**a**, **b**)?
 - 1) $n\%2019==0 \ \&\& \ n<a \ || \ n>b$
 - 2) $!(n\%2019 \ || \ !(n<=a \ \&\& \ n>=b))$

- 3) $\text{floor}(a) \geq \text{abs}(n) \ \&\& \ !(\text{abs}(n) < \text{ceil}(b) \ || \ n \% 2019 != 0)$
 4) $a \geq n \ || \ b \leq n \ \&\& \ (n \% 3 == 0 \ \&\& \ n \% 673 == 0)$
3. Un număr natural este palindrom dacă este egal cu oglinditul său (De exemplu: 202, 1881 sunt numere palindrom, iar 2012 nu este palindrom). Știind că în variabila x este memorat un număr natural de exact 5 cifre, stabilește care dintre următoarele expresii C/C++ are valoarea 1 dacă și numai dacă numărul x este palindrom.
- a) $x \% 100 == x / 100 / 10$
 b) $x / 10 \% 10 + x \% 10 * 10 == x / 100 / 10$
 c) $x / 1000 \% 10 == x / 100 \% 10 \ \&\& \ x / 10000 == x \% 10$
 d) $x / 10000 == x / 10 \% 10 \ \&\& \ x / 1000 == x \% 100$
4. Știind că x este o variabilă care memorează un număr întreg nenul, precizează care este cea mai mică valoare ce se poate obține la evaluarea următoarei expresii C/C++: $200 \% x - x \% 5$.
- a) -4 b) 0 c) -203 d) -195
5. Care este rezultatul ce se obține la evaluarea expresiei C/C++ de mai jos?
 $(5 * 4 \% 7 - 19 / (7 - 4 \% 5)) / 2$
- a) 1 b) 0 c) 1.5 d) 1.1416
6. Se consideră următoarea secvență de instrucțiuni C-C++. Precizează câte operații de atribuire se vor efectua la execuția acesteia.
- ```
int n = 1023, k = 2;
while (n > 0 || k > 0)
{
 n = n / 10;
 k = k - 1;
}
```
- a) 10                      b) 6                      c) 8                      d) 12
7. Se consideră  $x$  și  $y$  variabile care memorează numere naturale și următoarele 3 instrucțiuni de atribuire în C/C++. Precizează care este ordinea în care trebuie executate acestea, astfel încât să se realizeze interschimbarea valorilor celor două variabile.
- 1)  $x = y - x$                       2)  $y = x + y$                       3)  $y = y - x$                       4)  $x = x - y$   
 a) 2, 3, 1                      b) 1, 2, 3                      c) 3, 1, 2                      d) 3, 2, 1
8. Știind că  $a$ ,  $x$  și  $y$  sunt variabile reale nenule, scrie instrucțiunea C/C++ corespunzătoare următoarei operații de atribuire:  $a \leftarrow \frac{(x+y^2) \cdot (y-1)}{x^2 \cdot y}$ .
9. Știind că variabilele  $x$ ,  $y$ ,  $z$  memorează numere naturale, nenule și distincte două câte două, care dintre următoarele expresii C/C++ au valoarea 1 dacă și numai dacă numărul memorat în variabila  $z$  este fie multiplu de  $x$  și de  $y$ , fie este divizor al lui  $x$  și al lui  $y$ ?
- a)  $!(z \% x \ || \ z \% y) \ \&\& \ x \% z == 0 \ \&\& \ z \% y == 0$   
 b)  $(z \% x + z \% y == 0) \ || \ !(x \% z \ || \ y \% z)$   
 c)  $(x \% z == 0 \ || \ y \% z == 0) \ \&\& \ y \% x == 0 \ \&\& \ z \% y == 0$   
 d)  $z \% (x * y) == 0 \ || \ (x * y) \% z == 0$

## CAPITOLUL 2



### Algoritmi elementari

#### Teorie



#### I. Algoritmi care prelucreză cifrele unui număr

- spargerea în cifre a numărului  $n$  și prelucrarea cifrelor de la dreapta la stânga:

|                                                                                                                                                                                                                             |                                                                                                                                                                 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> dacă <math>n=0</math> atunci     prelucrare_cifra(0) ■ cât timp <math>n \neq 0</math> execută     <math>cif \leftarrow n \% 10</math>     prelucrare_cifra(cif)     <math>n \leftarrow [n/10]</math> ■         </pre> | <pre> repetă     <math>cif \leftarrow n \% 10</math>     prelucrare_cifra(cif)     <math>n \leftarrow [n/10]</math> ■ până când <math>n=0</math>         </pre> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|

- spargerea în cifre a numărului  $n$  și prelucrarea cifrelor de la stânga la dreapta:

|                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> <math>p \leftarrow 1</math> cât timp <math>p * 10 \leq n</math> atunci     <math>p \leftarrow p * 10</math> ■ cât timp <math>p \neq 0</math> execută     <math>cif \leftarrow [n/p]</math>     prelucrare_cifra(cif)     <math>n \leftarrow [n \% p]</math>     <math>p \leftarrow [p/10]</math> ■         </pre> | <p>Formarea numărului cu cifrele din <math>n</math> de pe poziții impare, considerând numerotarea de la stânga la dreapta, prima cifră fiind considerată pe poziția 1.</p> <pre> <math>p \leftarrow 1</math> cât timp <math>p * 10 \leq n</math> atunci     <math>p \leftarrow p * 10</math> ■ <math>nrNou \leftarrow 0</math> cât timp <math>p \neq 0</math> execută     <math>cif \leftarrow [n/p]</math>     <math>nrNou \leftarrow nrNou * 10 + cif</math>     <math>n \leftarrow [n \% (p * 10)]</math>     <math>p \leftarrow [p/100]</math> ■         </pre> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

prelucrare\_cifra(cif) – descrie operațiile specifice problemei în care apare această prelucrare. De exemplu: suma cifrelor/numărarea cifrelor cu o anumită proprietate, verificarea proprietății de palindrom, eliminarea/inserarea cifrelor cu o anumită proprietate etc.

**Atenție!** Spargerea în cifre a unui număr distruge valoarea inițială a acestuia! Este necesară crearea unei copii a lui  $n$ , anterior spargerii, pentru a putea procesa ulterior valoarea inițială.

# Exemple

| Construirea oglinditului și<br>verificare $n$ dacă este palindrom                                                                                                                                                                                         | Media aritmetică<br>a cifrelor nenule                                                                                                                                                                                                                                                                             | Cifra maximă<br>din $n$                                                                                                                                                                                                 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> copie ← n oglindit ← 0 cât timp <math>n \neq 0</math> execută     cif ← <math>n \% 10</math>     oglindit ← <math>oglindit * 10 + cif</math>     <math>n \leftarrow [n/10]</math> dacă copie=oglindit atunci     prelucrare_cifra(palindrom) </pre> | <pre> sum ← 0; nrcif ← 0; medie ← 0; cât timp <math>n \neq 0</math> execută     cif ← <math>n \% 10</math>     dacă <math>cif &gt; 0</math> atunci         sum ← sum + cif         nrcif ← nrcif + 1     <math>n \leftarrow [n/10]</math> dacă nrcif &gt; 0 atunci     medie ← sum/nrcif prelucrare(medie) </pre> | <pre> cifmax ← 0 cât timp <math>n \neq 0</math> execută     cif ← <math>n \% 10</math>     dacă <math>cif &gt; cifmax</math> atunci         cifmax ← cif     <math>n \leftarrow [n/10]</math> prelucrare(cifmax) </pre> |

| Eliminarea cifrelor<br>impare din $n$                                                                                                                                                                                                                  | Dublarea aparițiilor<br>cifrelor pare din $n$                                                                                                                                                                                                                                                     | Numărarea cifrelor<br>pare existente în $n$                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> nrNou ← 0 p ← 1 cât timp <math>n \neq 0</math> execută     cif ← <math>n \% 10</math>     dacă <math>cif \% 2 = 0</math> atunci         nrNou ← nrNou + cif * p         p ← p * 10     <math>n \leftarrow [n/10]</math> prelucrare(nrNou) </pre> | <pre> nrNou ← 0 p ← 1 cât timp <math>n \neq 0</math> execută     cif ← <math>n \% 10</math>     dacă <math>cif \% 2 = 0</math> atunci         nrNou ← nrNou + cif * p         p ← p * 10     nrNou ← nrNou + cif * p     p ← p * 10     <math>n \leftarrow [n/10]</math> prelucrare(nrNou) </pre> | <pre> nrPare ← 0 repetă     cif ← <math>n \% 10</math>     dacă <math>cif \% 2 = 0</math> atunci         nrPare ← nrPare + 1     <math>n \leftarrow [n/10]</math> până când <math>n = 0</math> prelucrare(nrPare) </pre> |

**Cifra de control** a unui număr  $n$  se obține calculând suma cifrelor lui  $n$  apoi repetând procesul cu cifrele sumei obținute anterior până când se obține un număr format dintr-o singură cifră, numită cifră de control. De exemplu, pentru  $n=7912$  se obțin pe rând sumele  $7+9+1+2=19$ ,  $1+9=10$ ,  $1+0=1$ , iar 1 este cifra de control a lui 7912.

```

cât timp $n > 9$ execută
 sumcif $\leftarrow 0$
 cât timp $n \neq 0$ atunci
 cif $\leftarrow n \% 10$
 sumcif $\leftarrow$ sumcif + cif
 $n \leftarrow [n/10]$
 $n \leftarrow$ sumcif
prelucrare(n)

```

Algoritmul eficient ca timp de execuție se bazează pe observația că cifra de control a unui număr este periodică și respectă relația:

$$\text{cifControl}(n) = \begin{cases} 0, & \text{dacă } n=0 \\ 9, & \text{dacă } n\%9=0 \text{ și } n \neq 0 \\ n\%9, & \text{altfel} \end{cases}$$

## II. Divizibilitate. Algoritmi care prelucrează divizorii proprii/improprii/primi ai unui număr

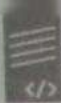
Fie  $n$  număr natural. Definim mulțimile:

Divizorii **proprii**:  $d \in \{2, 3, 4, \dots, [\frac{n}{2}]\}, n : d$

Divizorii **primi**:  $d \in \{2, 3, 5, 7 \dots [\sqrt{n}]\}, n : d$

Divizorii **improprii**:  $d \in \{1, n\}$

| Divizorii proprii ai lui $n$                                                                                                                                                                                                                                                                                                                     | Divizorii proprii ai lui $n$ – optimizat                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> p <math>\leftarrow 1</math> pentru <math>d \leftarrow 2, [n/2], 1</math> execută     dacă <math>n \% d = 0</math> atunci         prelucrare (d) </pre>                                                                                                                                                                                     | <pre> p <math>\leftarrow 1</math> pentru <math>d \leftarrow 2, [\sqrt{n}] - 1, 1</math> execută     dacă <math>n \% d = 0</math> atunci         prelucrare (d)         prelucrare (n/d)     dacă <math>d * d = n</math> atunci         prelucrare (d) </pre>                                                                                                                                                         |
| Divizorii primi                                                                                                                                                                                                                                                                                                                                  | Divizorii primi ai lui $n$ – optimizat                                                                                                                                                                                                                                                                                                                                                                               |
| <pre> d <math>\leftarrow 2</math> cât timp <math>n &gt; 1</math> execută     p <math>\leftarrow 0</math>     cât timp <math>n \% d = 0</math> execută         n <math>\leftarrow [n/d]</math>         p <math>\leftarrow p + 1</math>     dacă p <math>\neq 0</math> atunci         prelucrare (d, p)     d <math>\leftarrow d + 1</math> </pre> | <pre> d <math>\leftarrow 2</math> cât timp <math>d * d \leq n</math> execută     p <math>\leftarrow 0</math>     cât timp <math>n \% d = 0</math> execută         n <math>\leftarrow [n/d]</math>         p <math>\leftarrow p + 1</math>     dacă p <math>\neq 0</math> atunci         prelucrare (d, p)     d <math>\leftarrow d + 1</math>     dacă <math>n \neq 1</math> atunci         prelucrare (n, 1) </pre> |



Numere cu proprietăți speciale:

- $N$  are număr impar de divizori  $\Leftrightarrow N$  este pătrat perfect (de exemplu,  $N = 36$ );
- $N$  are exact 3 divizori  $\Leftrightarrow N$  este pătrat perfect de număr prim (de exemplu,  $N = 25$ );
- $N$  este număr perfect  $\Leftrightarrow N$  este egal cu suma divizorilor mai mici decât  $N$ ;
- $A$  și  $B$  sunt numere prietene  $\Leftrightarrow A$  este egal cu suma divizorilor lui  $B$ , mai mici decât  $B$ , iar  $B$  este egal cu suma divizorilor lui  $A$ , mai mici decât  $A$ .

*Formula lui Euler* pentru calcularea numărului de divizori ai lui  $n$ , descompus în factori prim sub forma:

$$n = p_1^{f_1} p_2^{f_2} \dots p_k^{f_k}, \text{ unde } p_1, p_2, \dots, p_k \text{ sunt factorii primi la puterile } f_1, f_2, \dots, f_k$$

$$\text{nrDivizori}(n) = (f_1 + 1)(f_2 + 1) \dots (f_k + 1)$$

### III. Primalitate. Testarea primalității unui număr $n$

|                                                                                                                                                                                                    |                                                                                                                                                                                  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>prim ← 1    //presupunem că n e prim dacă n &lt; 2 atunci     prim ← 0 pentru d ← 2, [√n], 1 execută     dacă n % d = 0 atunci         prim ← 0 dacă prim = 1 atunci     prelucrare (n)</pre> | <pre>prim ← 1 d ← 2 cât timp prim = 1 și d * d ≤ n execută     dacă n % d = 0 atunci         prim ← 0     altfel         d ← d + 1 dacă prim = 1 atunci     prelucrare (n)</pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### IV. Cel mai mare divizor comun. Cel mai mic multiplu comun

|                                                                              |                                                                                                           |
|------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| <pre>cât timp b ≠ 0 execută     rest ← a % b     a ← b     b ← rest</pre>    | <pre>cât timp b ≠ a execută     dacă a &gt; b atunci         a ← a - b     altfel         b ← b - a</pre> |
| Cmmdc (a, b) – Algoritmul lui Euclid bazat pe împărțiri succesive (EFICIENT) | Cmmdc (a, b) – Algoritmul lui Euclid bazat pe scăderi succesive                                           |

$Cmmmc(a, b)$  se poate determina folosind formula de calcul:

$$Cmmmc(a, b) = \frac{a * b}{cmmdc(a, b)}$$

Numerele  $a$  și  $b$  sunt **prime între ele**  $\Leftrightarrow cmmdc(a, b) = 1$

## V. Șiruri recurente

Generarea termenilor din șirul Fibonacci, definit de relațiile:

$$f_0 = f_1, f_n = \begin{cases} 1, & \text{dacă } n = 1 \text{ sau } n = 0 \\ f_{n-1} + f_{n-2}, & \text{dacă } n > 1 \end{cases}$$

|                                                                                                                                     |                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> F0 ← 1 F1 ← 1 Prelucrare(F0,F1)   pentru i ← 2, n, 1 execută     F2 ← F1+F0     Prelucrare(F2)     F0 ← F1     F1 ← F2 </pre> | <pre> F0 ← 1 F1 ← 1   cât timp n &gt; 0 execută     F2 ← F1+F0     dacă F2%2 = 0 atunci       scrie F2       n ← n-1     F0 ← F1     F1 ← F2 </pre> |
| Prelucrarea primilor $n$ termeni din șirul Fibonacci                                                                                | Primii $n$ termeni Fibonacci pari                                                                                                                   |

## VI. Baze de numerație. Conversii între baza 10 și baza $b$ , $2 \leq b \leq 9$ , pentru un număr $n$

|                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Conversie din baza 10 în baza <math>b</math></b>, realizată prin împărțiri succesive la <math>b</math>, cât timp <math>n &gt; 0</math>. Resturile, în ordine inversă obținerii lor, formează numărul în baza <math>b</math>, egal cu numărul <math>n</math>, în baza 10.</p> <p>De exemplu: <math>75_{10} = 1001011_{(2)}</math></p> | <p><b>Conversie din baza <math>b</math> în baza 10</b>, realizată prin dezvoltare după puterile bazei <math>b</math>. Numărul <math>n</math> se sparge în cifre, care se înmulțesc cu puterile corespunzătoare ale bazei <math>b</math>.</p> <p>De exemplu: <math>1001011_{(2)} = 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2 + 1 = 75_{(10)}</math></p> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



|                                                                                     |                                                                                                                                                          |
|-------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>repetă   cif ← n % b   Prelucrare(cif)   n ← [n / b]   ■ până când n = 0</pre> | <pre>nrNou ← 0 p ← 1   cât timp n &gt; 0 execută     cif ← n % 10     nrNou ← nrNou + p * cif     n ← [n / 10]     p ← p * b   ■ Prelucrare(nrNou)</pre> |
|-------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|

Pentru baza  $b > 10$ , mulțimea de simboluri care reprezintă alfabetul bazei este  $\{0, 1, \dots, 9, 'A', 'B', \dots\}$ . Resturile obținute în algoritmul de conversie din baza 10 în baza  $b$ , sunt cuprinse în intervalul  $[0, b - 1]$ , iar cele mai mari decât 9 se vor înlocui cu litera corespunzătoare din alfabet.

Observație: Baza minimă în care un număr poate fi considerat corect reprezentat este cu 1 mai mare decât cifra minimă din număr.

#### Algoritmul de conversie din baza 10 în baza $b$ , $b > 10$

```
repetă
 rest ← n % b
 dacă rest < 10 atunci
 prelucrare ('0' + rest)
 altfel
 prelucrare ('A' + rest - 10)
 ■
 n ← [n / b]
 ■ până când n = 0
```

#### VII. Parcurgerea și prelucrarea numerelor dintr-un interval. Citirea și prelucrarea pe rând a $n$ numere

|                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                             |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Prelucrarea numerelor dintr-un interval <math>[a, b]</math>, <math>a \leq b</math> sau <math>[1, n]</math>, parcurse crescător</p> <pre>citește a, b   pentru <math>x \leftarrow a, b, 1</math> execută     prelucrare (x)   ■</pre> | <p>Prelucrarea a <math>n</math> numere citite pe rând</p> <pre>citește n   pentru <math>i \leftarrow 1, n, 1</math> execută     citește x     Prelucrare(x)   ■</pre> <p>Prelucrarea perechilor dintr-un șir de <math>n</math> valori citite consecutiv</p> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



|                                                                                                                                                                                                                                                                                                         |                                                                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| citește $n$<br>┌ pentru $x \leftarrow 1, n, 1$ execută<br>│ prelucrare ( $x$ )<br>└ ── ■                                                                                                                                                                                                                | citește $n, a$<br>┌ pentru $i \leftarrow 2, n, 1$ execută<br>│ citește $b$<br>│ Prelucrare( $a, b$ )<br>│ $a \leftarrow b$<br>└ ── ■ |
| Prelucrarea numerelor dintr-un interval<br>$[a, b], a \leq b$ sau $[1, n]$ , parcurse descrescător<br><br>citește $a, b$<br>┌ pentru $x \leftarrow b, a, -1$ execută<br>│ prelucrare ( $x$ )<br>└ ── ■<br><br>citește $n$<br>┌ pentru $x \leftarrow n, 1, -1$ execută<br>│ prelucrare ( $x$ )<br>└ ── ■ |                                                                                                                                      |

**Exemple de optimizări** în algoritmi de numărare a valorilor dintr-un interval, care au o anumită proprietate:

- 1) Numărul de multipli de  $k$ , mai mici sau egali decât  $n$ , este egal cu  $n/k \Rightarrow$  Numărul de multipli de  $k$ , din intervalul  $[a, b]$  este  $b/k - (a-1)/k$
- 2) Numărul de valori mai mici sau egale cu  $n$ , care au un număr impar de divizori, este  $[\sqrt{n}]$ , deoarece doar pătratele perfecte sunt valori cu număr impar de divizori  $\Rightarrow$  Numărul de valori din intervalul  $[a, b]$  care au un număr impar de divizori este  $[\sqrt{b}] - [\sqrt{a-1}]$ .
- 3) Numărul de valori  $x, x \leq n$  care au cifra de control  $k, 0 \leq k \leq 9$  este:

$$nr(n, k) = \begin{cases} 1, & \text{dacă } n = 0 \text{ și } k = 0 \\ 0, & \text{dacă } n = 0 \text{ sau } k = 0 \\ \left\lfloor \frac{n}{9} \right\rfloor, & \text{dacă } n \% 9 < k \\ \left\lfloor \frac{n}{9} \right\rfloor + 1, & \text{dacă } n \% 9 \geq k \end{cases}$$

- 4) Numărul de valori  $x, x \in [a, b]$  care au cifra de control  $k, 0 \leq k \leq 9$  este  $nr(b, k) - nr(a-1, k)$ .

**PROBLEME PROPUSE**

- |                                                                                                                                                                                                                               |                                                                           |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| 1. Se consideră programul pseudocod alăturat.<br>S-a notat cu $x \% y$ restul împărțirii numerelor întregi $x$ și $y$ și cu $[z]$ partea întreagă a numărului real $z$ .<br>a) Care este valoarea afișată pentru $a = 1372$ ? | citește $a$ (număr natural)<br>$i \leftarrow 0$<br>$a \leftarrow a \% 10$ |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|



- b) Scrie cea mai mică valoare de 3 cifre care se poate citi pentru  $a$ , astfel încât valoarea afișată de algoritmul alăturat să fie maximă.
- c) Scrie un algoritm echivalent cu algoritmul dat în care structura **cât timp... execută** să fie înlocuită cu o structură repetitivă cu test final.
- d) Scrie programul C/C++ corespunzător algoritmului dat.

2. Se consideră programul pseudocod alăturat. S-a notat cu  $[z]$  partea întreagă a numărului real  $z$ .

- a) Care este valoarea afișată la citirea următorului șir de valori 5 37 205 199 30 86?
- b) Scrie un șir de valori distincte, de cel mult două cifre pentru care algoritmul va afișa 9999.
- c) Scrie un algoritm echivalent cu algoritmul dat în care prima structura **cât timp... execută** să fie înlocuită cu o structură repetitivă **pentru**.
- d) Scrie programul C/C++ corespunzător algoritmului dat.

3. Se consideră programul pseudocod alăturat. S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[z]$  partea întreagă a numărului real  $z$ .

- a) Ce va afișa algoritmul alăturat dacă se citesc valorile  $a=12$   $b=24$ ?
- b) Scrie toate perechile de valori de câte o cifră care pot fi citite pentru  $a$  și  $b$ , astfel încât ultima pereche afișată de algoritmul alăturat să fie 6 7.
- c) Scrie un algoritm echivalent cu algoritmul dat în care prima structura **repetă... până când** să fie înlocuită cu o structură repetitivă cu condiție inițială.
- d) Scrie programul C/C++ corespunzător algoritmului dat.

4. Se consideră programul pseudocod alăturat. S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[z]$  partea întreagă a numărului real  $z$ .

- a) Ce va afișa algoritmul alăturat dacă se citesc valorile  $a=10011$  și  $b=2$ ?

```

cât timp ($a > 1$) și ($a < 10$) execută
 $i \leftarrow i + 1$
 $a \leftarrow a * a$
scrie $i * a$

```

citește  $n$  (număr natural nenul)

```

 $a \leftarrow 0$
cât timp $n > 0$ execută
 citește x (număr natural nenul)
 $p \leftarrow 1$
 cât timp $p \leq [x/10]$ atunci
 $p \leftarrow p * 10$
 $a \leftarrow a * 10 + [x/p]$
 $n \leftarrow n - 1$
scrie a

```

citește  $a, b$  (numere naturale nenule,  $a < b$ )

```

 $p \leftarrow a; u \leftarrow b$
cât timp $p < u$ execută
 $x \leftarrow p; y \leftarrow u$
 repetă
 $z \leftarrow x \% y$
 $x \leftarrow y$
 $y \leftarrow z$
 până când $y = 0$
 dacă $x = 1$ atunci
 scrie $p, ', u, ',$
 $p \leftarrow p + 1; u \leftarrow u - 1$

```

citește  $a, b$  (numere naturale  $> 0$ ,  $2 \leq b \leq 9$ )

$m \leftarrow 0; x \leftarrow 0$

- b) Dacă pentru  $b=8$ , ce valoare poate fi citită pentru  $a$ , astfel încât algoritmul alăturat să afișeze 559?
- c) Scrie un algoritm echivalent cu algoritmul dat care să utilizeze o singură structură repetitivă.
- d) Scrie programul C/C++ corespunzător algoritmului dat.

```

cât timp $a > 0$ execută
 $p \leftarrow a \% 10$
 pentru $i \leftarrow 1$, m execută
 $p \leftarrow p * b$
 $x \leftarrow x + p$
 $a \leftarrow [a/10]$
 $m \leftarrow m + 1$
scrie x

```

5. Se consideră programul pseudocod alăturat. S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[z]$  partea întreagă a numărului real  $z$ .

```

citește n, a (număr natural nenul)
 $x \leftarrow 0$
pentru $i \leftarrow 1$, n execută
 citește b (numere naturale)
 $0 < a \leq b$
 $p \leftarrow 1$
 cât timp $p < b$ execută
 $p \leftarrow p * a$
 $x \leftarrow x + [b/p]$
scrie x

```

- a) Ce va afișa algoritmul alăturat dacă se citesc valorile 5 2 32 110 4 243 512?
- b) Pentru  $n=3$ ,  $a=5$  scrie un șir de valori distincte, de cel mult două cifre, ce pot fi citite pentru  $b$ , astfel încât algoritmul alăturat să afișeze 3.
- c) Scrie un algoritm echivalent cu algoritmul dat în care structura **cât timp... execută** să fie înlocuită cu o structură repetitivă cu condiție finală.
- d) Scrie programul C/C++ corespunzător algoritmului dat.

6. Se consideră programul pseudocod alăturat. S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[z]$  partea întreagă a numărului real  $z$ .

```

citește x, y (numere naturale, $0 < x \leq y$)
 $k \leftarrow 0$
cât timp $y \geq x$ execută
 $d \leftarrow 2$
 cât timp $d * d \leq y$ și $y \% d \neq 0$ execută
 $d \leftarrow d + 1$
 dacă $d * d = y$ atunci
 scrie $y, ' '$
 $k \leftarrow k + 1$
 $y \leftarrow y - 1$
scrie k

```

- a) Ce va afișa algoritmul alăturat dacă se citesc valorile  $x=2$  și  $y=10$ ?
- b) Scrie o pereche de valori de două cifre ce pot fi citite pentru  $x$  și  $y$ , astfel încât algoritmul alăturat să afișeze la sfârșit valoarea 1.
- c) Scrie un algoritm echivalent cu algoritmul dat în care prima structură **cât timp... execută** să fie înlocuită cu o structură **pentru... execută**.
- d) Scrie programul C/C++ corespunzător algoritmului dat.



7. Se consideră programul pseudocod alăturat. S-a notat cu  $x\%y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[z]$  partea întreagă a numărului real  $z$ .

a) Ce va afișa algoritmul alăturat dacă se citește  $n=120$ ?

b) Scrie cea mai mică valoare de trei cifre ce poate fi citită pentru  $n$ , astfel încât algoritmul alăturat să afișeze o singură valoare, pentru fiecare dintre acestea.

c) Scrie un algoritm echivalent cu algoritmul dat în care a doua structură **cât timp...** **execută** să fie înlocuită cu o structură repetitivă cu condiție finală.

d) Scrie programul C/C++ corespunzător algoritmului dat.

8. Se consideră programul pseudocod alăturat. S-a notat cu  $x\%y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[z]$  partea întreagă a numărului real  $z$ .

a) Ce va afișa algoritmul alăturat dacă se citește  $n=32153$ ?

b) Câte numere de cel mult două cifre pot fi citite pentru  $n$ , astfel încât algoritmul alăturat să afișeze 1?

c) Scrie un algoritm echivalent cu algoritmul dat în care structura **cât timp...** **execută** să fie înlocuită cu o structură repetitivă cu condiție finală.

d) Scrie programul C/C++ corespunzător algoritmului dat.

9. Se consideră programul pseudocod alăturat. S-a notat cu  $x\%y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[z]$  partea întreagă a numărului real  $z$ .

a) Ce va afișa algoritmul alăturat dacă se citește  $n=4200$ ?

b) Scrie două valori distincte, de cel puțin trei cifre, ce pot fi citite pentru  $n$  astfel încât algoritmul alăturat să scrie o valoare egală cu  $n$ , pentru fiecare dintre ele.

c) Scrie un algoritm echivalent cu algoritmul dat în care a doua structură **cât timp...** **execută** să fie înlocuită cu o structură repetitivă cu condiție finală.

d) Scrie programul C/C++ corespunzător algoritmului dat.

citește  $n$  (număr natural nenule,  $2 \leq n$ )  
 $k \leftarrow 0$

**cât timp**  $n > 0$  **execută**

$a \leftarrow 1; b \leftarrow 0$

**cât timp**  $b \leq n$  **execută**

$b \leftarrow a + b$

$a \leftarrow b - a$

scrie  $a$ , "

$n \leftarrow n - a$

citește  $n$  (număr natural)

$p \leftarrow 1; x \leftarrow 0;$

**cât timp**  $p < [y/p]$  **execută**

$x \leftarrow x * 10 + [n/p] \% 10$

$p \leftarrow p * 10$

**dacă**  $x = [n/p]$  sau  $x = [n/(p/10)]$   
sau  $n < 10$

atunci scrie 1

altfel scrie 0

citește  $n$  (număr natural nenul,  $n > 1$ )

$x \leftarrow 2; p \leftarrow 1$

**cât timp**  $x \leq n$  **execută**

$k \leftarrow 0;$

**cât timp**  $n \% x = 0$  **execută**

$n \leftarrow [n/x]$

$k \leftarrow k + 1$

**dacă**  $k \% 2 \neq 0$  **atunci**

$p \leftarrow p * x$

$x \leftarrow x + 1$

scrie  $p$



## CAPITOLUL 3



### Fișiere text

#### Teorie



**Fișierul** este o colecție de date omogene memorate pe un suport extern. Fișierul se identifică printr-un nume și o extensie. Într-un fișier text datele sunt memorate sub forma unei succesiuni de caractere (memorate prin utilizarea codului ASCII). Fișierul text este format din una sau mai multe linii, o linie se termină cu caracterul newline, mai puțin ultima care se termină cu marcajul de sfârșit de fișier (CTRL+Z).

#### Lucrul cu fișiere text

Pentru a lucra cu fișiere text trebuie să includem header-ul `fstream`.

```
#include<fstream>
```

Fiecărui fișier fizic îi corespunde în program un fișier logic (o variabilă), care trebuie declarată înaintea utilizării ei. Dacă un fișier text este utilizat numai pentru operații de citire acesta poate fi deschis astfel:

```
ifstream nume_logic(„nume_fizic”);
```

Dacă un fișier text este utilizat numai pentru operații de scriere acesta poate fi deschis astfel:

```
ofstream nume_logic(„nume_fizic”);
```

#### Citirea și afișarea datelor

Se consideră declarațiile următoare:

```
ifstream fin(„bac.in”);
```

```
ofstream fout(„bac.out”);
```

```
int x;
```

```
//citirea variabilei x din fisier
```

```
fin>>x;
```

```
//afișarea variabilei x în fișierul de ieșire
```

```
fout<<x;
```

#### Citirea datelor până la sfârșitul fișierului

Presupunem ca avem enunțul: Fișierul **bac.txt** conține valori întregi, acestea trebuie prelucrate, secvența <prelucrează x> realizează prelucrările dorite, aceasta va fi modificată conform cerințelor. Prelucrarea datelor dintr-un fișier când nu se cunoaște numărul de valori din fișier se poate face după un algoritm similar cu unul dintre algoritmii următori.

```
a) ifstream fin(„bac.txt”);
```

```
int x;
```

```
while(fin>>x)
```

```
{<prelucrează x>}
```

```
b) ifstream fin("bac.txt");
 int x;
 fin >> x;
 while (!fin.eof())
 { <prelucrează x>
 fin >> x;
 }
```

Testarea sfârșitului de fișier se poate face folosind funcția `eof()` care nu citește, ci doar testează dacă anterior a fost detectat sfârșitul de fișier. Apelul funcției se face astfel: `nume_logic.eof()`.

#### Închiderea fișierelor text

```
nume_logic.close();
```

#### PROBLEME PROPUSE

1. Fișierul `text numere.in` conține pe prima linie un număr natural  $N$ , iar pe următoarele  $N$  linii câte o pereche de numere naturale  $a$  și  $b$ . Să se scrie un program care citește perechile de valori din fișier și afișează în fișierul `numere.out` pe linii diferite, pentru fiecare pereche din fișier cel mai mare divizor comun și cel mai mic multiplu comun al numerelor din pereche. Numerele din fișier au cel mult patru cifre.
2. Fișierul `text numere.in` conține pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 9 cifre fiecare. Să se scrie un program care să determine și să afișeze pe ecran cele mai mici trei valori din fișier care sunt multipli ai numărului 3.
3. Fișierul `text numere.in` conține pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 9 cifre fiecare, în ordine crescătoare. Să se scrie un program care să afișeze pe ecran pe aceeași linie, cu spațiu între ele, valorile distincte din fișier.  
**Exemplu:** Dacă fișierul `bac.txt` conține numerele 12 12 31 31 31 31 23 atunci pe ecran se afișează valorile 12 31 23.
4. Numim **secvență10**, o secvență de numere aflate pe poziții consecutive în fișier cu proprietatea că toate numerele din secvență sunt numere divizibile cu 10. Fișierul `text numere.in` conține pe prima linie un număr natural  $N$ , iar pe următoarea linie un șir cu  $N$  numere naturale. Să se scrie un program care să se determine și să afișeze pe ecran lungimea maximă a secvențelor din fișier care sunt **secvență10**. Pentru determinarea valorii cerute se utilizează un algoritm eficient din punct de vedere al memoriei utilizate și al timpului de executare.
5. Fișierul `text bac.txt` conține pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 9 cifre fiecare. Să se scrie un program care citește valorile din fișier și afișează pe ecran pe aceeași linie cu spațiu între ele valorile din fișier care sunt puteri ale lui 2.
6. Să se scrie un program C++ care citește de la tastatură două numere naturale  $a$  și  $b$ , proprietatea că au toate cifrele egale.  
**Exemplu:** Dacă  $a = 82$  și  $b = 567$  în fișier se vor afișa 88 99 111 222 333 444 555.

7. Să se scrie un program C++ care citește de la tastatură două numere naturale  $a$  și  $b$ , ( $a < b < 10^9$ ) și afișează în fișierul `bac.out` toate numerele din intervalul  $[a, b]$  cu proprietatea că împărțite la suma cifrelor lor dau restul egal cu câtul.

**Exemplu:** Dacă  $a = 10$  și  $b = 200$  în fișier se vor afișa **16 39 55 96 187**.

8. Să se scrie un program C++ care citește de la tastatură un număr natural  $n$  și afișează în fișierul `bac.out` pe linii diferite toate prefixele numărului  $n$ .

**Exemplu:** Dacă  $a = 823451$  în fișier se vor afișa pe linii diferite **823451 82345 8234 823 82 8**.

9. Fișierul `text numere.in` conține pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 9 cifre fiecare. Să se scrie un program care să determine și să afișeze pe ecran separate printr-un spațiu ultima valoare din fișier care este multiplu al numărului 5 și numărul său de ordine.

**Exemplu:** Dacă fișierul `bac.txt` conține numerele 12 13 35 123 325 61 25 54 425 511, atunci pe ecran se afișează valorile **425 9**.



## CAPITOLUL 4



## Tablouri unidimensionale (vectori)

## Teorie



Tabloul este un tip de date structurat în care elementele sunt de același tip, numit tip de bază al tabloului. Un tablou unidimensional se numește **vector**.

## Declarare

```
tip nume[nr_max_elem];
```

Unde: `tip` este tipul elementelor tabloului; `nr_max_elem` este o expresie constantă ce reprezintă numărul maxim de elemente din tablou.

## Exemplu:

```
int v[100], n, i;
```

Am declarat un vector cu 100 de elemente de tip întreg, putem utiliza doar  $n$  elemente din vector,  $n \leq 100$ , spunem că  $n$  este numărul efectiv de elemente din vector.

## Citirea și afișarea elementelor unui tablou

```
int v[100], n, i;
```

```
//citirea elementelor
```

```
cin>>n;
```

```
for(i=0;i<n;i++)
```

```
cin>>v[i];
```

```
//afișarea elementelor pe aceeași linie/pe linii diferite
```

```
for(i=0;i<n;i++)
```

```
cout<<v[i]<<' ';//cout<<v[i]<<endl;
```

## Algoritmi elementari cu vectori

```
int v[100], n, i, k, aux, x;
```

## 1. Permutare circulară la stânga

```
aux=v[0];
```

```
for(i=1;i<n;i++)
```

```
v[i-1]=v[i];
```

```
v[n-1]=aux;
```



## 2. Permutare circulară la dreapta

```
aux=v[n-1];
for (i=n-2;i>=0;i--)
 v[i+1]=v[i];
v[0]=aux;
```

## 3. Inserarea unui nou element în poziția k din vector

```
for (i=n-1;i>=k;i--)
 v[i+1]=v[i];
v[k]=x;
n++;
```

## 4. Ștergerea elementului din poziția k din vector

```
for (i=k+1;i<n;i++)
 v[i-1]=v[i];
n--;
```

## 5. Sortarea elementelor unui vector

### a) Sortare prin interschimbare (metoda bulelor)

Vectorul se parcurge în mod repetat comparând două elemente consecutive ale vectorului și interschimbându-le în cazul neîndeplinirii criteriului de ordonare (în cazul ordonării crescătoare dacă  $v[i] > v[i+1]$ ). Astfel, elementele cu valoare mare sunt „împinse” către sfârșitul vectorului. Algoritmul se termină atunci când, la o parcurgere a vectorului nu s-a produs nicio interschimbare. Complexitatea algoritmului este  $O(n^2)$ . Considerăm că avem următoarea declarație de variabile:

```
int v[100], n, i, aux, sortat;
```

Secvența de program care ordonează crescător elementele vectorului prin metoda bulelor este:

```
do
{
 sortat = 1;
 for (i=0; i<n-1; i++)
 if (v[i] > v[i+1])
 {
 aux = v[i]; v[i] = v[i+1]; v[i+1] = aux;
 sortat = 0;
 }
} while (!sortat);
```

### b) Sortare prin selecția minimului (maximumului)

Sortarea prin selecție presupune determinarea elementului minim (maxim) din vector pentru fiecare poziție și plasarea acestuia pe poziția curentă, prin interschimbare. Complexitatea algoritmului este  $O(n^2)$ . Considerăm că avem următoarea declarație de variabile:

```
int v[100], n, i, aux, mini;
```



Secvența de program care ordonează crescător elementele vectorului prin metoda selecției minimului (analog se face și sortarea prin selecția maximului) este:

```
for (i=0; i<n-1; i++)
{
 mini = v[i];
 poz_min = i;
 for (j=i+1; j<n; j++)
 if (v[j] < mini)
 {
 mini = v[j];
 poz_min = j;
 }
 v[poz_min] = v[i];
 v[i] = mini;
}
```

c) *Sortare prin selecție directă*

```
for (i=0; i<n-1; i++)
 for (j=i+1; j<n; j++)
 if (v[i]>v[j])
 {
 aux = v[i]; v[i] = v[j]; v[j] = aux;
 }
```

## 6. Algoritmi de căutare

a) *Căutare liniară (secvențială)*

Operația de căutare constă în parcurgerea secvențială a vectorului și testarea egalității dintre elementul curent și valoarea căutată. Considerăm că avem următoarea declarație de variabile:

```
int v[100], n, i, x, poz;
```

Putem identifica două cazuri:

a.1. Elementele vectorului sunt într-o ordine oarecare, în această situație căutarea se oprește când am găsit elementul (căutare cu succes) sau când am epuizat elementele din vector (căutare fără succes).

```
poz = 0;
while ((x != v[poz]) && (poz < n))
 poz++;
if (poz < n)
 cout << x << " se afla pe pozitia " << poz;
else
 cout << x << " nu se gaseste in vectorul v";
```

a.2. Elementele vectorului sunt ordonate crescător (descrescător), în această situație căutarea se oprește când am găsit elementul (căutare cu succes) sau când am găsit primul element mai mare strict (mai mic strict) decât elementul căutat (căutare fără succes).

```

poz = 0;
while ((x < v[poz]) && (poz < n))
 poz++;
if (v[poz]==x)
 cout << x << "se afla pe pozitia " << poz;
else
 cout << x << "nu se gaseste in vectorul v";

```

#### b) Căutare binară

Vectorul în care se face căutarea trebuie fie ordonat. Ideea algoritmului: căutarea se efectuează printre valorile reținute de elementele de indice ls (limita stângă) și ld (limita dreaptă). Inițial ls = 0 și ld = n - 1. Procedeu constă în:

- se determină indicele elementului din mijloc  $mij = (ls + ld) / 2$

Apar trei situații:

- dacă  $x = v[mij]$  atunci căutarea se termină cu succes;
- dacă  $x < v[mij]$  atunci valoarea va fi căutată între elementele de indice ls și mij - 1;
- dacă  $x > v[mij]$  atunci valoarea va fi căutată între componentele de indice mij + 1 și ld.

Secvența de program care implementează algoritmul de căutare binară este:

```

ls=1;ld=n;ok=0;
while(ls<=ld&&!ok)
{
 mij=(ls+ld)/2;
 if(v[mij]==d)ok=1;
 else if(d<v[mij])ld=mij-1;
 else ls=mij+1;
}
if(ok)
 cout<<x<<" se afla in sir in pozitia "<<mij;
else
 cout<<x<<" trebuie inserat pe pozitia "<<ls<<" astfel v
 ramane ordonat".

```

### 7. Interclasarea a doi vectori ordonați

Interclasarea presupune obținerea din doi vectori ordonați (crescător sau descrescător) a unui singur vector ordonat. Considerăm că avem următoarea declarație de variabile:

```
int a[100], n, i, b[100], m, j, c[200], k;
```

Ideea algoritmului:

- se compară primul element din vectorul a cu primul din b și pe cel mai mic îl trecem în vectorul c, după care înaintăm în vectorul din care am preluat elementul;
- procesul se repetă până când se termină unul dintre vectori;
- în final se copiază la sfârșitul lui c toate elementele din vectorul rămas, acestea fiind ordonate.

Secvența de program care implementează algoritmul de interclasare este:

```
i = j = k = 0;
while ((i < n) && (j < m))
{
 if (a[i] < b[j])
 c[k++] = a[i++];
 else
 c[k++] = b[j++];
}
// daca au mai ramas elemente in a
for (; i < n; i++)
 c[k++] = a[i];
// daca au mai ramas elemente in b
for (; j < m; j++)
 c[k++] = b[j];
```

#### 8. Vector de frecvență/Vector caracteristic/vector de poziție

- Fiind dat un șir cu foarte multe elemente, valorile din șir fiind din mulțimea  $\{0, 1, 2, \dots, v_{\max}\}$ , pentru a memora valorile din șir și a le prelucra în ordinea crescătoare (descrescătoare) a valorilor din șir putem utiliza un vector cu  $v_{\max}+1$  elemente, în care un element  $vf[i]$  reprezintă numărul de apariții (frecvența) a valorii  $i$  în șir, în acest caz particular avem un algoritm de ordonare de complexitate liniară  $O(n)$ .
- Dacă ne interesează doar dacă o valoare a apărut sau nu în șir, putem utiliza un vector care memorează doar valorile 1 sau 0 după cum valoarea  $i$  aparține sau nu șirului.
- Sunt situații în care trebuie să memorăm poziția primei/ultimei apariții a valorii  $i$  în șir.

#### PROBLEME PROPUSE

1. În fișierul `numere.in` se află pe prima linie,  $N$ , un număr natural par, iar pe următoarele linii, se află  $N$ , numere naturale, reprezentând elementele unui vector, valorile din fișier au cel mult 4 cifre. Să se scrie un program care citește valorile din fișier și afișează pe ecran vectorul obținut prin permutarea circulară la stânga cu o poziție a elementelor din prima jumătate și prin permutarea circulară la dreapta cu o poziție a elementelor din a doua jumătate.
2. Să se scrie un program C++, care citește de la tastatură un număr natural  $N$ , cu cel mult 10 cifre, programul determină și afișează cel mai mare număr care se poate obține utilizând:
  - a) toate cifrele lui  $N$ ;
  - b) toate cifrele distincte ale lui  $N$ .
3. În fișierul `numere.in` se găsesc pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 2 cifre fiecare. Să se scrie un program care citește valorile din fișier și le afișează în fișierul `numere.out` în ordine crescătoare. Se va utiliza un algoritm eficient din punct de vedere al timpului de executare.



4. În fișierul `numere.in` se găsesc pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 2 cifre fiecare. Să se scrie un program care citește valorile din fișier și afișează în fișierul `numere.out` în ordine crescătoare valorile distincte din fișierul `numere.in`. Se va utiliza un algoritm eficient din punct de vedere al timpului de executare.
5. În fișierul `numere.in` se găsesc pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 2 cifre fiecare. Să se scrie un program care citește valorile din fișier și afișează în fișierul `numere.out` pe linii diferite toate valorile distincte din fișierul `numere.in` urmate de distanța dintre prima și ultima apariție, dacă o valoare apare o singură dată, distanța va fi 0, se va utiliza un algoritm eficient din punct de vedere al timpului de executare.
6. În fișierul `numere.in` se află pe prima linie,  $N$  ( $N < 500$ ), un număr natural, iar pe următoarele linii, se află  $N$ , numere naturale, reprezentând elementele unui vector. Să se scrie un program care citește valorile din fișier și afișează pe ecran vectorul obținut prin inserarea valorii 2019 după fiecare element par.
7. În fișierul `numere.in` se află pe prima linie,  $N$ , un număr natural, iar pe următoarele linii, se află  $N$ , numere naturale, reprezentând elementele unui vector. Să se scrie un program care citește valorile din fișier și elimină din vector un număr minim de elemente astfel încât în vector să nu se afle două valori egale pe poziții consecutive. Vectorul astfel obținut se va afișa pe ecran.
8. Fișierul `bac.txt` conține pe prima linie numărul natural  $N$  ( $N < 500$ ), pe a doua linie un șir de  $N$  numere întregi, cu cel mult 4 cifre fiecare, ordonat strict crescător, iar pe a treia linie un număr  $X$  de același tip cu elementele vectorului. Numerele de pe aceeași linie sunt separate prin câte un spațiu. Se cere să se afișeze pe ecran cel mai mare număr din șir mai mic sau egal cu  $X$ . Dacă nu există un astfel de număr, se afișează pe ecran mesajul `nu exista`. Pentru determinarea numărului cerut se utilizează un algoritm eficient din punct de vedere al timpului de executare.
9. Fișierul `bac.txt` conține pe prima linie două numere naturale,  $M$  și  $N$  ( $M, N < 500$ ), iar pe fiecare dintre următoarele două linii câte un șir de  $M$ , respectiv  $N$  numere naturale ordonate crescător. Numerele aflate pe aceeași linie a fișierului sunt separate prin câte un spațiu. Se cere să se afișeze pe ecran, în ordine descrescătoare, toate numerele impare aflate în cele două șiruri. Numerele afișate sunt separate prin câte un spațiu, iar dacă nu există niciun astfel de număr, se afișează pe ecran mesajul `nu există`. Pentru determinarea numerelor cerute se va utiliza un algoritm eficient din punct de vedere al timpului de executare.



## CAPITOLUL 5



## Tablouri bidimensionale (matrice)

## Teorie



Tabloul este un tip de date structurat în care elementele sunt de același tip, numit tip de bază al tabloului. Un tablou bidimensional se numește **matrice**.

## Declarare

```
tip nume[nr_max_linii] [nr_max_coloane];
```

Unde: `tip` este tipul elementelor tabloului; `nr_max_linii` este o expresie constantă ce reprezintă numărul maxim de linii din matrice, `nr_max_coloane` este o expresie constantă ce reprezintă numărul maxim de coloane din matrice.

## Exemplu:

```
int a[10][10], L, C, i, j;
```

Am declarat o matrice cu 10 linii și 10 coloane și elemente de tip întreg, putem utiliza doar `L` linii din matrice și `C` coloane din matrice,  $L \leq 10$ ,  $C \leq 10$ . Spunem că `L` este numărul efectiv de linii din matrice și `C` este numărul efectiv de coloane din matrice.

## Citirea și afișarea elementelor unei matrice

```
int a[10][10], L, C, i, j;
//citirea elementelor
cin>>L>>C;
for(i=0;i<L;i++)
 for(j=0;j<C;j++)
 cin>>a[i][j];
//afișarea elementelor matricei
for(i=0;i<L;i++)
 {for(j=0;j<C;j++)
 cout<<a[i][j]<<' ';
 cout<<endl;
}
```

## Matrice pătratică

Se numește matrice pătratică, o matrice în care numărul de linii este egal cu numărul de coloane ( $L = C$ ). Într-o matrice pătratică avem cele două diagonale, diagonala principală și diagonala secundară.



## Diagonala principală

int A[6][6], L, i, j;

- Pentru  $L = 5$  și numerotarea liniilor și coloanelor de la 1 la L, elementele de pe diagonala principală sunt evidențiate în matricea de mai jos.
- Pe diagonala principală se află elementele de forma  $A[i][j]$ , cu proprietatea că  $i = j$ , sau elementele  $A[i][i]$ , unde  $i = 1, L$  (pentru numerotarea liniilor și coloanelor de la 1 la L) sau  $i = 0, L - 1$  (pentru numerotarea liniilor și coloanelor de la 0 la  $L - 1$ ).
- **Sub diagonala principală** se află elementele  $A[i][j]$ , cu  $i > j$ .
- **Deasupra diagonalei principale** se află elementele  $A[i][j]$ , cu  $i < j$ .

|         |         |         |         |         |
|---------|---------|---------|---------|---------|
| A[1][1] |         |         |         |         |
|         | A[2][2] |         |         |         |
|         |         | A[3][3] |         |         |
|         |         |         | A[4][4] |         |
|         |         |         |         | A[5][5] |

## Diagonala secundară

int A[6][6], L, i, j;

### Cazul 1: Numerotarea liniilor și coloanelor de la 1 la L

- Pentru  $L = 5$  și numerotarea liniilor și coloanelor de la 1 la L, elementele de pe diagonala secundară sunt evidențiate în matricea de mai jos.
- Pe diagonala secundară se află elementele de forma  $A[i][j]$ , cu proprietatea că  $i + j = L + 1$ , sau elementele  $A[i][L + 1 - i]$ , unde  $i = 1, L$ .
- **Sub diagonala secundară** se află elementele  $A[i][j]$ , cu  $i + j > L + 1$ .
- **Deasupra diagonalei secundare** se află elementele  $A[i][j]$ , cu  $i + j < L + 1$ .

|         |         |         |         |         |
|---------|---------|---------|---------|---------|
|         |         |         |         | A[1][5] |
|         |         |         | A[2][4] |         |
|         |         | A[3][3] |         |         |
|         | A[4][2] |         |         |         |
| A[5][1] |         |         |         |         |

### Cazul 2: Numerotarea liniilor și coloanelor de la 0 la $L - 1$

- Pentru  $L = 5$  și numerotarea liniilor și coloanelor de la 0 la  $L - 1$ , elementele de pe diagonala secundară sunt evidențiate în matricea de mai jos.
- Pe diagonala secundară se află elementele de forma  $A[i][j]$ , cu proprietatea că  $i + j = L - 1$ , sau elementele  $A[i][L - 1 - i]$ , unde  $i = 0, L - 1$ .
- **Sub diagonala secundară** se află elementele  $A[i][j]$ , cu  $i + j > L - 1$ .
- **Deasupra diagonalei secundare** se află elementele  $A[i][j]$ , cu  $i + j < L - 1$ .

|         |         |         |         |         |
|---------|---------|---------|---------|---------|
|         |         |         |         | A[0][4] |
|         |         |         | A[1][3] |         |
|         |         | A[2][2] |         |         |
|         | A[3][1] |         |         |         |
| A[4][0] |         |         |         | 1       |

### Vecinii unui element

Într-o matrice, în funcție de poziția sa, un element poate avea 3, 5 sau 8 vecini. Pentru un element  $A[i][j]$ , vecinii săi sunt reprezentați în matricea următoare.

|               |             |               |
|---------------|-------------|---------------|
| $A[i-1][j-1]$ | $A[i-1][j]$ | $A[i-1][j+1]$ |
| $A[i][j-1]$   | $A[i][j]$   | $A[i][j+1]$   |
| $A[i+1][j-1]$ | $A[i+1][j]$ | $A[i+1][j+1]$ |

### PROBLEME PROPUSE

1. Scrie un program C/C++ care citește de la tastatură un număr natural  $N$  ( $2 < N \leq 20$ ) și o valoare întreagă  $x$ , programul construiește în memorie un tablou bidimensional cu  $N$  linii și  $N$  coloane în care: toate elementele din prima linie și prima coloană au valoarea  $x$ ; oricare alt element este obținut prin însumarea celor două elemente vecine cu el, aflate pe linia anterioară și pe aceeași coloană cu el, respectiv pe aceeași linie cu el și pe coloana anterioară, ca în exemplu. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu.  
**Exemplu:** Pentru  $n = 4$  și  $x = 1$  pe ecran se afișează tabloul de mai jos.

```
1 1 1 1
1 2 3 4
1 3 6 10
1 4 10 20
```

2. Scrie un program C/C++ care citește de la tastatură două valori naturale nenule  $M$  și  $N$  ( $M \leq 10$ ,  $N \leq 10$ ) și apoi  $M \cdot N$  numere naturale nenule cu cel mult 4 cifre fiecare, reprezentând elementele unei matrice cu  $M$  linii și  $N$  coloane. Programul determină apoi valorile maxime de pe fiecare linie a matricei și afișează pe ecran suma valorilor maxime determinate.

**Exemplu:** Pentru  $M = 3$ ,  $N = 4$  și matricea

```
32 71 3 5
10 12 6 9
```

74 55 6 3 se afișează pe ecran valoarea 157 ( $157 = 71 + 12 + 74$ ).

3. Scrie un program C/C++ care citește de la tastatură două valori naturale nenule  $M$  și  $N$  ( $M \leq 10$ ,  $N \leq 10$ ) și apoi  $M \cdot N$  numere naturale nenule cu cel mult 4 cifre fiecare, reprezentând elementele unei matrice cu  $M$  linii și  $N$  coloane. Programul determină apoi valorile minime de pe fiecare coloană a matricei și afișează pe ecran suma valorilor minime determinate.

**Exemplu:** Pentru  $M = 3$ ,  $N = 4$  și matricea

32 71 3 5

10 12 6 9

74 55 6 3 se afișează pe ecran valoarea 28 ( $28 = 10 + 12 + 3 + 3$ ).

4. Scrie un program C/C++ care citește de la tastatură un număr natural,  $N$  ( $N < 10$ ), apoi  $N$  numere naturale din intervalul  $[0, 10]$ , reprezentând valorile elementelor aflate pe prima linie a unui tablou bidimensional cu  $N$  linii și  $N$  coloane. Programul construiește în memorie tabloul, inițializând celelalte elemente, astfel încât fiecare linie să se obțină prin permutarea circulară a elementelor liniei anterioare spre stânga, cu o poziție, ca în exemplu. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele de pe aceeași linie fiind separate prin câte un spațiu.

**Exemplu:** Dacă se citesc numerele  $n = 4$ , 1 2 3 4 apoi se obține tabloul:

1 2 3 4

2 3 4 1

3 4 1 2

4 1 2 3

5. Scrie un program C/C++ care citește de la tastatură numere naturale din intervalul  $[2, 100]$ , în această ordine:  $N$ , apoi elementele unui tablou bidimensional cu  $N$  linii și  $N$  coloane, iar la final un număr  $x$ . Programul afișează pe ecran numărul valorilor aflate pe conturul tabloului (format din prima linie, ultima linie, prima coloană și ultima coloană) care sunt multipli ai numărului  $x$ .

**Exemplu:** Pentru  $N = 4$ , tabloul următor și  $x = 4$  se va afișa 4.

5 2 4 3

8 2 4 1

6 4 1 5

8 4 5 2

6. Scrie un program C/C++ care citește de la tastatură două numere naturale  $M$  și  $N$  ( $2 \leq M \leq 50$ ,  $2 \leq N \leq 50$ ) și elementele unui tablou bidimensional cu  $M$  linii și  $N$  coloane, numere naturale cu cel mult patru cifre fiecare. Programul determină liniile care au toate elementele egale cu aceeași valoare și, pentru fiecare astfel de linie afișează pe ecran pe linii diferite numărul liniei separat printr-un spațiu de valoarea respectivă, iar dacă nu există astfel de valori, programul afișează pe ecran mesajul Nu există.

**Exemplu:** Pentru  $M=3$ ,  $N=4$  și tabloul alăturat, se afișează pe ecran numerele 1 5, iar pe linia următoare 3 8.

5 5 5 5

8 2 4 1

8 8 8 8

7. Într-un tablou bidimensional format numai din valorile 0 sau 1, spunem că două coloane sunt complementare, dacă oricare două elemente aflate pe aceeași linie sunt diferite. Scrie un program C/C++ care citește de la tastatură două numere naturale  $M$  și  $N$  ( $2 \leq M \leq 50$ ,  $2 \leq N \leq 50$ ) și elementele unui tablou bidimensional cu  $M$  linii și  $N$  coloane, numere naturale

din mulțimea  $\{0,1\}$ , programul afișează pe ecran câte coloane complementare cu prima coloană sunt în matrice.

**Exemplu:** Pentru  $M = 3$ ,  $N = 4$  și tabloul alăturat, se afișează pe ecran 2.

```
0 1 1 1
1 1 0 0
1 1 0 0
```

8. Scrie un program C/C++ care citește de la tastatură un număr natural  $N$  ( $2 < N \leq 15$ ) și construiește în memorie un tablou bidimensional cu  $N$  linii și  $N$  coloane în care toate valorile naturale din intervalul  $[1, N*N]$  sunt așezate șerpuit ca în exemplu. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu.

**Exemplu:** Pentru  $n = 4$  pe ecran se afișează tabloul de mai jos.

```
1 2 3 4
8 7 6 5
9 10 11 12
16 15 14 13
```

9. Scrie un program C/C++ care citește de la tastatură un număr natural  $N$  ( $2 < N \leq 15$ ) și construiește în memorie un tablou bidimensional cu  $N$  linii și  $N$  coloane în care toate valorile naturale din intervalul  $[1, N*N]$  sunt așezate în spirală ca în exemplu. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu.

**Exemplu:** Pentru  $n = 4$  pe ecran se afișează tabloul de mai jos.

```
1 2 3 4
12 13 14 5
11 16 15 6
10 9 8 7
```

## CAPITOLUL 6



## Șiruri de caractere

## Teorie



**Pointer** este o variabilă ce memorează adresa unei zone de memorie și se declară astfel:

**tip\_data \*nume\_pointer;**

Un **șir de caractere** reprezintă o succesiune de caractere ASCII, care se încheie cu caracterul `'\0'` (NULL).

## Declarare șir de caractere

- a) `char nume_sir[lungime_maximă];` lungime\_maxima reprezintă numărul maxim de caractere din șir +1.  
 b) `char *nume_sir;`

## Exemple

`char s[101],*q;`

La declarare, un șir de caractere poate fi și inițializat: `char t[5]="abc".`

|      |      |      |      |
|------|------|------|------|
| a    | b    | c    | NULL |
| t[0] | t[1] | t[2] | t[3] |

## Citirea unui șir de caractere

`char s[lungime_maximă];`

- a) Dacă șirul nu conține spații se poate utiliza:

`cin>>s;`

- b) Dacă șirul conține spații se utilizează una dintre variantele următoare:

1. `cin.get(s, lungime_maxima, caracter_final);`

`caracter_final` este implicit caracterul newline `'\n'`.

Se citește șirul s până la întâlnirea caracterului final (fără să-l introducă în șir) sau până la citirea unui număr maxim de lungime\_maxima-1 caractere și adaugă NULL la sfârșitul șirului.

2. `cin.getline(s, lungime_maxima, caracter_final);`

Se citește șirul s până la întâlnirea caracterului final sau până la citirea unui număr maxim de lungime\_maxima-1 caractere, adaugă NULL la sfârșitul șirului și extrage `caracter_final` din fluxul de intrare (il va omite la citirea următoare din program).

## Afișarea unui șir de caractere

`cout<<s;`

# TEZE

Specializarea Științe ale Naturii

# Teza 1

## SUBIECTUL I

(20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele  $a$ ,  $b$  și  $z$  sunt reale, iar  $a \leq b$ . Care dintre expresiile C/C++ următoare are valoarea 1 dacă și numai dacă valoarea variabilei  $z$  aparține intervalului închis determinat de valorile variabilelor  $a$  și  $b$ ?

- a)  $(z > a) \mid (z > b)$   
b)  $(z < a) \mid (z > b)$   
c)  $z < a \ \&\& \ z > b$   
d)  $z \geq a \ \&\& \ z \leq b$

$z \in [a, b]$

2. Aplicând algoritmul de căutare binară pentru căutarea unei valori  $x$  în tablourile unidimensionale  $(1, 5, 7, 10, 20)$  și  $(19, 10, 9, 8, 7, 6, 5)$  se fac două comparații în ambele cazuri. Care dintre următoarele valori poate fi valoarea variabilei  $x$ ?

- a) 5                      b) 10                      c) 8                      d) 6

3. Indicați expresia care are valoarea 1 dacă și numai dacă numărul memorat în variabila întreagă  $x$  este număr natural.

- a)  $\text{floor}(x) == \text{abs}(x)$   
b)  $\text{floor}(x) == \text{ceil}(x)$   
c)  $\text{floor}(x) - 1 == x$   
d)  $\text{abs}(x) == x$

4. Variabilele  $i$  și  $j$  sunt de tip întreg. Indicați expresia care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine.

```
for (i=1; i<=5; i++)
{
 for (j=1; j<=5; j++)
 cout<<...<<' '
 cout<<endl;
}
```

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

- a)  $i+j$   
b)  $(i+j) \% 5$   
c)  $(i+j-1) \% 5$   
d)  $i+j-5$



5. Se consideră algoritmul:  
 citește x (x număr natural)  
 $y \leftarrow 0$

```

 cât timp x > 0 execută
 [
 $y \leftarrow y + x \% 10$
 $x \leftarrow [x/100]$
]
 ■

```

Ce se va afișa dacă se citește pentru x valoarea 14360?

a) 2

b) 1

c) 3

d) 4

## SUBIECTUL II

(40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi x și y și cu  $[x]$  partea întreagă a numărului real x.

- a) Scrieți valoarea care se va afișa pentru x = 2019.

(6 puncte)

- b) Scrieți cel mai mic număr de două cifre și cel mai mare număr de două cifre care pot fi citite pentru x astfel încât rezultatul afișat să fie 9.

(6 puncte)

- c) Scrieți programul C/C++ corespunzător algoritmului dat.

(10 puncte)

- d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se nu se utilizeze structuri repetitive.

(6 puncte)

citește x  
 (x număr natural)

```

 cât timp x > 9 execută
 [
 $y \leftarrow 0$
 cât timp x > 0 execută
 [
 $y \leftarrow y + x \% 10$
 $x \leftarrow [x/10]$
]
 ■
 $x \leftarrow y$
]
 ■
 scrie x

```

2. Se consideră declarațiile de mai jos, în care variabila nr memorează numărul de vârfuri ale unui poligon (număr natural din intervalul  $[3, 10^2]$ ) și tablourile x și y memorează coordonatele vârfurilor acestuia (abscisa și ordonata) în sistemul de coordonate xOy (numere reale). Fără a utiliza variabile suplimentare scrieți o secvență de instrucțiuni C/C++ care determină și afișează pe ecran perimetrul poligonului memorat în variabila per.

```
int i, nr;
```

```
float per, x[101], y[101];
```

(6 puncte)

3. În urma utilizării algoritmului de interclasare descrescătoare a două tablouri unidimensionale cu elemente distincte se obține tabloul (20, 19, 10, 10, 9, 8, 7, 5, 5, 1, 1). Dați exemplu de două astfel de tablouri ordonate crescător astfel încât să se obțină rezultatul precizat.

(6 puncte)

1. Se citește un număr natural  $n$  și se cere să se scrie suma divizorilor primi ai lui  $n$ .

**Exemplu:** Dacă  $n = 12$ , atunci se scrie 5, iar dacă  $n = 11$  atunci se scrie 11. Scrieți, în pseudocod, algoritmul de rezolvare al problemei enunțate.

(10 puncte)

2. Scrieți un program C/C++ care citește de la tastatură numărul natural  $n$  ( $n \in [2, 10]$ ), apoi cele  $n$  elemente ale unui tablou unidimensional, numere naturale din intervalul  $[0, 10^2]$ , și afișează pe ecran, separate printr-un spațiu, primul număr și ultimul număr memorate în tablou, apoi pe linia următoare primele două numere din tablou și ultimele două numere din tablou, ș.a.m.d. în final primele  $n$  numere din tablou și ultimele  $n$  numere din tablou ca în exemplu.

(10 puncte)

**Exemplu:** Dacă  $n=4$  și elementele tabloului sunt 2 4 7 1 atunci pe ecran se va afișa:

2 1

2 4 7 1

2 4 7 4 7 1

2 4 7 1 2 4 7 1

3. Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir  $S$  cu cel mult 100 000 de numere naturale formate din cel mult trei cifre fiecare. Asupra acestui șir se aplică în mod repetat următoarea prelucrare: se elimină din șir valorile prime, iar valorile neprime se incrementează cu valoarea 1. Prelucrarea se repetă până când în șir nu mai rămâne niciun număr. Să se afișeze pe ecran de câte ori a fost aplicată această prelucrare.

**Exemplu:** Dacă fișierul conține numerele 12 11 16 45 34 atunci după prima prelucrare vom avea numerele 13 17 46 35; după a doua 47 36; după a treia 37; astfel pe ecran va fi afișată valoarea 4.

- a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de execuție și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia.

(2 puncte)

- b) Scrieți programul C/C++ corespunzător metodei descrise la punctul a).

(8 puncte)

## Teza 2

(20 de puncte)

## SUBIECTUL I

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

- Variabilele  $a, b$  sunt numere naturale. Care dintre expresiile C/C++ are valoarea 1 dacă și numai dacă valorile variabilelor  $a$  și  $b$  sunt numere naturale consecutive?
  - $a-b==1$
  - $a==1 \& \& b==0$
  - $a-b==1 \& \& b-a==1$
  - $a-b==1 \mid \mid b-a==1$
- Aplicând algoritmul de căutare binară pentru căutarea unei valori  $x$  în tabloul unidimensional (19, 15, 11, 9, 8, 7, 1) se fac exact trei comparații. Care dintre următoarele valori poate fi valoarea variabilei  $x$ ?
  - 19, 11
  - 15, 7
  - 8, 1
  - 19, 11, 8, 1
- Variabilele  $n, k, nr$  memorează numere naturale. Variabila  $nr$  trebuie să memoreze numărul de numere din intervalul  $[0, n]$  care sunt divizibile cu  $k$ . Cu ce expresie trebuie completată atribuirea  $nr=...$ ?
  - $nr = n/k + 1$
  - $nr = n/k$
  - $nr = n/k + n\%k$
  - $nr = n\%k + 1$
- Variabilele  $i$  și  $j$  sunt de tip întreg. Indicați expresia care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine.

```
for (i=1; i<=5; i++)
{
 for (j=1; j<=5; j++)
 if (....)
 cout<<3<<' \';
 else
 cout<<1<<' \';
 cout<<endl;
}
```

```
1 1 1 1 3
1 1 1 3 1
1 1 3 1 1
1 3 1 1 1
3 1 1 1 1
```

- $i+j==5$
- $i==j$
- $i+j==6$
- $i\%2==j\%2$

5. Se consideră algoritmul:

$x \leftarrow 3$  (x număr natural)

[ pentru  $i \leftarrow 1, 100$  execută

dacă  $x < 8$  atunci  $x \leftarrow x * 2$

scrie x

Ce se va afișa în urma execuției algoritmului?

a) 12

b) 32

c) 36

d) 24

## SUBIECTUL II

(40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

a) Scrieți valoarea care se va afișa dacă se citesc pe rând valorile 4, 8, 6, 16, 45. (6 puncte)

b) Pentru  $n = 4$ , scrieți un set de date de astfel încât rezultatul afișat să fie 0. (6 puncte)

c) Scrieți programul C/C++ corespunzător algoritmului dat. (10 puncte)

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se utilizeze alt tip de structură repetitivă. (6 puncte)

citește n (n număr natural)

$y \leftarrow 0$

[ cât timp  $n > 0$  execută

citește x

[ cât timp  $x \% 2 = 0$  execută

$x \leftarrow x / 2$

[ dacă  $x = 1$  atunci

$y \leftarrow y + 1$

$n \leftarrow n - 1$

scrie y

2. Două puncte M și N din planul  $xOy$  sunt date prin coordonatele lor cartezene  $XM, YM$  respectiv  $XN, YN$ . Scrieți o expresie C++ care are valoarea 1 dacă și numai dacă cele două puncte aparțin aceleiași axe de coordonate. (6 puncte)

3. Se consideră tablourile unidimensionale  $A = (3, 8, 15, 78, 80)$  și  $B = (15, 12, 9, 7, 3)$ . Scrieți elementele tabloului C, în ordinea în care ele apar în tablou, astfel încât acesta să fie obținut prin interclasarea descrescătoare a elementelor din A și B. (6 puncte)

(30 de puncte)

## SUBIECTUL III

1. Se citesc două numere naturale, a și b, se cere să se determine numărul de valori din intervalul închis determinat de a și b, care au exact 3 divizori. (10 puncte)

**Exemplu:** Dacă  $a = 3$  și  $b = 20$ , atunci se scrie 2 (doar 4 și 9 au exact 3 divizori). Scrieți în pseudocod, algoritmul de rezolvare al problemei enunțate.



2. Scrieți un program C/C++ care citește de la tastatură numărul natural  $n$ , unde  $n \in [2, 50]$ , apoi cele  $n$  elemente ale unui tablou unidimensional, numere naturale din intervalul  $[0, 10^9]$ , și afișează pe ecran pe linii diferite elementele distincte din tablou urmate de numărul lor de apariții. (10 puncte)

**Exemplu:** Dacă  $n=5$  și elementele tabloului sunt 2 4 7 4 7 atunci pe ecran se va afișa, nu neapărat în această ordine:

```
2 1
4 2
7 2
```

3. Din fișierul **bac.txt** se citesc  $n$  și  $m$  (numere naturale,  $0 < m < n < 100000$ ) de pe prima linie, apoi  $n$  numere naturale cu cel mult două cifre fiecare  $a_1, a_2, \dots, a_n$  de pe linia a doua și apoi  $m$  numere naturale cu cel mult două cifre fiecare  $b_1, b_2, \dots, b_m$  de pe linia a treia a fișierului. Să se determine câte șiruri  $b$  se pot obține din șirul  $a$  dacă se poate schimba ordinea elementelor din șirul  $a$ . Se va afișa pe ecran numărul numărul de șiruri obținute.

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. (2 puncte)

b) Să se scrie programul C/C++ ce realizează prelucrarea descrisă și afișează pe ecran un mesaj corespunzător. (8 puncte)

**Exemplu:** Pentru fișierul **bac.txt** cu conținutul:

8 3

1 6 3 1 3 7 6 1

6 1 3

se afișează valoarea 2.

## Teza 3

### SUBIECTUL I

(20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Valoarea variabilei  $a$  este un număr natural. Care dintre expresiile C/C++ următoare are valoarea 1 dacă și numai dacă valoarea variabilei  $a$  are exact 3 cifre?  
a)  $a/1000==0$   
b)  $a/100!=0$   
c)  $a/1000==0 \&\& a/100!=0$   
d)  $a/100==0 \&\& a/10!=0$
2. Se consideră două tablouri unidimensionale  $A$  și  $B$ . Dacă  $A = (2, 7, 9, 11, 18)$ , iar în urma interclasării lor în ordine descrescătoare se obține tabloul cu elementele  $(19, 18, 17, 12, 11, 9, 7, 7, 6, 2, 1)$ , atunci  $B$  poate fi:  
a)  $(19, 18, 7, 6, 1)$   
b)  $(19, 17, 12, 9, 7, 6, 2, 1)$   
c)  $(19, 17, 12, 7, 6, 1)$   
d)  $(19, 17, 12, 9, 7, 6, 1)$
3. Variabilele  $a, b, n$  memorează numere naturale. Care dintre expresiile C/C++ următoare are valoarea 1 dacă și numai dacă rădăcina pătrată a lui  $n$  nu depășește valoarea variabilei  $a$  și nici valoarea variabilei  $b$ ?  
a)  $\text{sqrt}(n) \leq a \&\& \text{sqrt}(n) \geq b$   
b)  $\text{sqrt}(n) \leq a \&\& \text{sqrt}(n) \leq b$   
c)  $\text{sqrt}(n) \leq a \mid \mid \text{sqrt}(n) \leq b$   
d)  $\text{sqr}(n) \leq a \&\& \text{sqr}(n) \leq b$
4. Se consideră un tablou unidimensional în care elementele sunt, în această ordine 50, 49, 47, 23, 21, 17, 12, 10, 7, 5, 3. Pentru a afla indicele elementului din tablou cu valoarea  $x = 49$ , se aplică metoda căutării binare. Scrieți succesiunea corectă de elemente a căror valoare se compară cu valoarea lui  $x$  pe parcursul metodei indicate.  
a) 17, 47, 50, 49  
b) 17, 47, 49  
c) 17, 50, 49  
d) 50, 49, 47
5. Se consideră algoritmul:  

```

┌ pentru i ← 2, 10 execută
│ dacă i%2=0 atunci
│ scrie 1
│ altfel
│ scrie 2
└
```

Ce se va afișa în urma execuției algoritmului?

a) 121212121

b) 212121212

c) 2121212121

d) 121212121



## SUBIECTUL II

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

- Scrieți valoarea care se va afișa pentru  $x = 7172$ . (6 puncte)
- Scrieți două seturi distincte de valori pentru  $x$  astfel încât rezultatul afișat să fie 4332. (6 puncte)
- Scrieți programul C/C++ corespunzător algoritmului dat. (10 puncte)
- Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura pentru cu alt tip de structură repetitivă. (6 puncte)

```

citește x
(x număr întreg)
y ← 0
pentru i ← 9, 0, -1 execută
 cx ← x
 cât timp cx > 0 execută
 dacă cx % 10 = i atunci
 y ← y * 10 + i
 cx ← cx / 10
scrie y

```

- Se consideră declarațiile de mai jos, în care variabila `pre` memorează partea reală și variabila `pim` memorează partea imaginară a unui număr complex  $z$ . Scrieți o secvență de instrucțiuni C/C++ care citește de la tastatură informațiile despre numărul complex  $z$  și afișează pe ecran modulul numărului complex  $z$ . (6 puncte)

```
int pre, pim;
```

- Variabilele  $i$  și  $j$  sunt de tip întreg. Scrieți expresia C/C++ care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. (6 puncte)

```
for (i=1; i<=5; i++)
 for (j=1; j<=5; j++)

```

```

2 1 1 1 1
1 2 1 1 1
1 1 2 1 1
1 1 1 2 1
1 1 1 1 2

```

## SUBIECTUL III

(30 de puncte)

- Se citesc două numere naturale,  $n$  și  $m$ , se cere să se determine numărul de cifre utilizate pentru construirea numerelor naturale din intervalul  $[n, m]$ .  
Exemplu: Dacă  $n = 7$  și  $m = 56$  atunci se va afișa valoarea 97. Scrieți, în pseudocod, algoritmul de rezolvare al problemei enunțate. (10 puncte)
- Scrieți un program C/C++ care citește de la tastatură numărul natural  $n$  ( $n \in [2, 100]$ ), apoi cele  $n$  elemente ale unui tablou unidimensional, numere naturale din intervalul  $[0, 10^9]$  știind

că elementele tabloului sunt ordonate crescător, să se elimine din vector un număr minim de elemente, astfel încât să nu existe în tablou două elemente alăturate identice. Să se afișeze pe ecran tabloul astfel obținut.

**Exemplu:** Dacă  $n = 7$  și elementele tabloului sunt 2 4 4 4 7 7 1 atunci pe ecran se va afișa 2 4 7 1.

(10 puncte)

3. Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir cu cel mult 100 000 de numere întregi formate din cel mult 2 cifre fiecare și afișează pe ecran separate printr-un spațiu, numărul sau numerele din fișier cu număr maxim de apariții.

**Exemplu:** Dacă în fișier sunt numerele 17 -8 13 17 -8 9 10 13 atunci pe ecran se vor afișa nu neapărat în această ordine -8 13 17.

- a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de execuție și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. (2 puncte)
- b) Scrieți programul C/C++ corespunzător metodei descrise la punctul a). (8 puncte)



## Teza 4

(20 de puncte)

## SUBIECTUL I

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele  $x$ ,  $y$  și  $z$  au valori numere naturale de cel mult 4 cifre. Care dintre expresiile C/C++ următoare are valoarea 1 dacă și numai dacă exact una dintre valori este impară?

- a)  $x\%2 \ || \ y\%2 \ || \ z\%2$   
b)  $(x+y)\%2==0 \ || \ (x+z)\%2==0 \ || \ (y+z)\%2==0$   
c)  $(x\%10 + y\%10 + z\%10)\%2>0$   
d)  $x\%10\%2 + y\%10\%2 + z\%10\%2==1$

2. Indicați cel mai mic și cel mai mare număr natural pe care îl poate memora variabila  $x$ , astfel încât expresia `sqrt(x+10) > x-10` să aibă valoarea 1.

- a) 6 și 10                      b) 0 și 14                      c) 7 și 10                      d) 0 și 10

3. Înlocuiți punctele de suspensie din secvența program C++ următoare, în ordinea scrierii lor în secvență, astfel încât să afișeze valoarea 1 dacă tabloul  $x$  format din  $n$  elemente este ordonat strict descrescător sau valoarea 0 în caz contrar.

```
int x[50], n, ordonat, i;
ordonat=...
for (i=2; i<=n; i++)
 if (...)
 {ordonat=0; break;}
if (...)
 cout<<1;
else
 cout<<0;
```

- a) 0,  $x[i] \leq x[i+1]$ ,  $ordonat==1$                       c) 1,  $x[i-1] < x[i]$ ,  $ordonat=1$   
b) 1,  $x[i] > x[i-1]$ ,  $ordonat==0$                       d) 1,  $x[i-1] \leq x[i]$ ,  $ordonat==1$

4. Pentru tablourile unidimensionale  $x$  care are 3 elemente și  $y$  care are 4 elemente, se aplică operațiile următoare:

- interclasarea tablourilor  $x$  și  $y$  prin care se obține tabloul  $z$ ;
- căutarea binară a valorii  $t$  în tabloul  $z$ . Valoarea  $t$  a fost găsită în tabloul  $z$  la a treia comparație, dacă s-au utilizat:

- a)  $x = (5, 9, 24)$ ,  $y = (2, 14, 7, 10)$ ,  $t = 7$                       c)  $x = (2, 7, 14)$ ,  $y = (5, 9, 10, 24)$ ,  $t = 30$   
b)  $x = (2, 7, 14)$ ,  $y = (5, 9, 10, 24)$ ,  $t = 30$                       d)  $x = (2, 7, 14)$ ,  $y = (5, 9, 10, 24)$ ,  $t = 24$

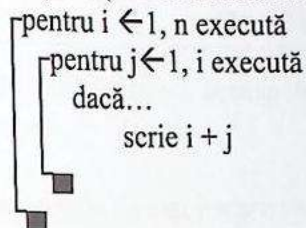
5. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

Completați punctele de suspensie astfel încât dacă se citește numărul 5, prin execuția acestui algoritm să se afișeze șase valori.

- a)  $i = j$       b)  $i \% 2 = 0$       c)  $i \% 2 = 1$       d)  $i \neq j$

citește  $n$  ( $n$  număr natural nenul)



## SUBIECTUL II

(40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

- a) Scrieți valoarea care se va afișa pentru  $n = 39$ . (6 puncte)
- b) Scrieți toate valorile de o cifră ce pot fi date pentru  $n$  astfel încât rezultatul afișat să fie 1. (6 puncte)
- c) Scrieți programul C/C++ corespunzător algoritmului dat. (10 puncte)
- d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se utilizeze cel mult o structură repetitivă. (6 puncte)

citește  $n$  ( $n$  număr natural nenul)

$a \leftarrow 1$

$b \leftarrow [n / 2]$

cât timp  $a \neq 0$  și  $b > 0$  execută

$c \leftarrow n$

cât timp  $c \geq b$  execută

$c \leftarrow c - b$

$a \leftarrow c$

$b \leftarrow b - 1$

$b \leftarrow b + 1$

scrie  $b$

2. Pentru memorarea rezultatelor la examenul de BAC obținute de 200 de candidați se consideră declarările C++ de mai jos:

```
int mat[200], lrom[200], inf[200];
```

Fiecare candidat are un cod de la 0 la 199, astfel candidatul cu codul  $i$  are la matematică nota  $mat[i]$ , la limba română nota  $lrom[i]$  și la informatică nota  $inf[i]$ . Scrieți o secvență de instrucțiuni C++ care determină și afișează numărul de candidați care au cel puțin nota 5 la fiecare probă și media generală cel puțin 7. (6 puncte)

3. În secvența de instrucțiuni de mai jos variabilele  $i$  și  $j$  sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze caracterele de mai jos, în această ordine. (6 puncte)

```
for (i=0; i<=6; i++) {
 for (j=0; j<=6; j++)

```

```
* - - - - - *
* * - - - * *
* - * - * - *
* - - * - - *
* - - - - - *
* - - - - - *
* - - - - - *
```

## SUBIECTUL III

(30 de puncte)

1. Un număr  $n$  se numește **rotund** dacă este par și în reprezentarea sa matematică în baza 2, nu sunt două cifre egale alăturate. De exemplu, numărul 10 este un număr rotund deoarece reprezentarea sa în baza 2 este 1010. Numărul 18 nu este rotund deoarece reprezentat în baza 2 este 10010. Scrieți un program în limbaj pseudocod care citește de la tastatură un număr natural nenul  $n$  cu cel mult 9 cifre și afișează valoarea 1 dacă  $n$  este număr rotund și 0 altfel. (10 puncte)

2. Scrieți un program C++ care citește de la tastatură un număr natural  $n$  ( $2 \leq n \leq 10^3$ ), afișează pe ecran un număr  $k$  ce reprezintă numărul de valori prime cu  $n$ , mai mici decât  $n$  și un tablou unidimensional  $p$  ce conține cele  $k$  valori prime cu  $n$ , în ordine crescătoare. Spunem că un număr  $x$  este prim cu  $n$  dacă nu au divizori comuni mai mari decât 1.

**Exemplu:** Dacă  $n=20$  atunci  $k=8$ , iar  $p=(1, 3, 7, 9, 11, 13, 17, 19)$ . (10 puncte)

3. Din fișierul text BAC.TXT se citesc, de pe prima linie un număr  $k$  ( $n < 10^6$ ), iar de pe linia următoare cel mult 1 000 000 de numere naturale cu cel mult 2 cifre fiecare, separate prin câte un spațiu. Să se afișeze pe ecran numărul de pe a doua linie din fișier care va fi pe poziția  $k$  în șirul ordonat descrescător, dacă există sau -1 în caz contrar. Să se proiecteze un algoritm eficient din punct de vedere al timpului de executare.

**Exemplu:** Dacă fișierul conține numerele:

6

21 9 16 2 16 2 9 4 9 21 9 2 8 atunci pe ecran se va afișa 9.

- a) Realizați o descriere de 3-4 rânduri a algoritmului ales justificând eficiența acestuia. (2 puncte)
- b) Scrieți programul C/C++ corespunzător metodei descrise la punctul a). (8 puncte)

## SUBIECTUL I

(20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele  $a, b, c, d$  sunt întregi,  $a \leq b, c \leq d$ . Care dintre expresiile de mai jos are valoarea 1 dacă și numai dacă intersecția intervalelor  $(a, b)$  și  $[c, d]$  este vidă?  
 a)  $c \geq b \ \&\& \ a < d$   
 b)  $!(d \geq a \ || \ c \leq b)$   
 c)  $!(b > c \ \&\& \ a < d)$   
 d)  $a \geq d \ || \ b < c$
2. Indicați ce valoare are expresia C++ următoare:  $\text{abs}(x)/\text{pow}(y, 2) - \text{pow}(y, 3) * \text{pow}(x+y, 2)$ , dacă toate variabilele din expresie sunt egale cu -1.  
 a) -3                      b) 5                      c) 2                      d) -6
3. Variabila  $a$  memorează un număr natural care are exact 4 cifre. Care dintre expresiile C/C++ de mai jos are ca valoare numărul format din prima și ultima cifră a numărului memorat de  $a$ ?  
 a)  $a/10 + a\%100$                       b)  $a/1000 + a\%10$                       c)  $a/1000*10 + a\%10$                       d)  $a - a/10\%10$
4. Aplicând algoritmul de căutare binară pentru căutarea valorii  $x = 8$  în tabloul unidimensional  $(19, 15, 11, 9, 8, 7, 1)$  se fac mai multe comparații. Care dintre următoarele valori este valoarea numărului de comparații?  
 a) 5                      b) 2                      c) 4                      d) 3
5. Se consideră o secvență de program, în care  $s, p$  și  $a$  sunt variabile întregi. Completați punctele de suspensie astfel încât după execuția acestei secvențe variabilele  $s$  și  $p$  să fie egale.

```

s=0; p=1; a=2354;
while (a>0)
{if(...)
 s=s+a%10;
else
 p=p*(a%10);
 a=a/10;
}

```

- a)  $a\%2==0$                       b)  $a\%2\%10==0$                       c)  $a\%10==0$                       d)  $a\%2!=0$

## SUBIECTUL II

1. Se consideră algoritmul alăturat, descris în pseudocod.

- S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$ .
- Scrieți valoarea care se va afișa dacă se citesc pe rând valorile 4, 8, 16, 10, 24. (6 puncte)
  - Pentru  $n = 3$ , dați exemple de trei numere naturale, nenule și distincte ce pot fi citite în variabila  $x$ , pentru care algoritmul să scrie 1. (6 puncte)
  - Scrieți programul C/C++ corespunzător algoritmului dat. (10 puncte)
  - Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să înlocuiți structura repetă până când cu o structură repetitivă condiționată anterior. (6 puncte)

citește  $n$  ( $n$  număr natural)  
 $d \leftarrow 0$

pentru  $i \leftarrow 1, n$  execută

citește  $x$

dacă  $d=0$  atunci

$d \leftarrow x$

altfel

repetă

$r \leftarrow x \% d$

$x \leftarrow d$

$d \leftarrow r$

până când  $r = 0$

$d \leftarrow x$

scrie  $d$

2. Un număr de 100 de elevi au optat fiecare pentru unul dintre cercurile: informatică, pictură, robotică. Codul unui cerc este prima literă a denumirii cercului. Pentru evidența înscrierii la cercuri se utilizează un tablou unidimensional **cercuri** care memorează codul cercului pentru fiecare elev (**i** – informatică, **p** – pictură, **r** – robotică). Știind că **cercuri[0]** reprezintă opțiunea primului elev, scrieți o secvență de instrucțiuni C++ care determină numărul maxim de elevi care s-au înscris la un cerc. (6 puncte)

3. În secvența de instrucțiuni de mai jos, variabilele **i** și **j** sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. (6 puncte)

```
for (i=1; i<=6; i++) {
 for (j=1; j<=6; j++)

```

```
1 1 1 1 1 1
1 2 2 2 2 1
1 2 3 3 2 1
1 2 3 3 2 1
1 2 2 2 2 1
1 1 1 1 1 1
```

1. Scrieți un program în limbaj pseudocod care citește de la tastatură un număr natural  $n$  de cel mult 6 cifre și o cifră nenulă  $c$  și afișează numărul maxim care se poate obține inserând în  $n$  cifra  $c$ .

**Exemplu:** Dacă  $n = 57332$  și  $c = 4$  atunci, se va afișa numărul 574332. (10 puncte)

2. Se citește de la tastatură un număr natural  $n$  ( $2 \leq n \leq 1000$ ) și un tablou unidimensional  $a$ , care memorează un șir de  $n$  numere naturale, fiecare cu cel mult 2 cifre. Se elimină din tablou, pentru fiecare  $a_i$ ,  $1 \leq i \leq n$ , elementele egale cu acesta, situate pe poziții consecutive, astfel încât, tabloul să conțină la final, un număr maxim de elemente distincte, în ordinea în care au apărut în șirul  $a$ . Scrieți un program C++ care afișează numărul de elemente rămase după eliminare și șirul modificat.

**Exemplu:** Dacă  $n = 9$  și tabloul  $a$  memorează valorile 1 2 2 2 3 3 5 8 9 atunci, se va afișa

5

1 2 3 5 8

(10 puncte)

3. Numim **interval asociat** unui șir de numere, perechea de numere naturale  $a, b$  ( $a \leq b$ ) cu proprietatea că șirul este format din toate valorile naturale distincte, cuprinse între  $a$  și  $b$  și scrise în ordine crescătoare. De exemplu, șirul 4 5 6 7 8 are intervalul asociat 4 8, iar șirul 1 2 3 5 7 8 9 este format din trei subșiruri ale căror intervale asociate sunt 1 3, 5 5 și 7 9.
- Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir  $S$  cu cel mult 1 000 000 de numere naturale din intervalul  $[0, 10^3]$  și afișează pe ecran extremitățile intervalului care reprezintă intersecția tuturor intervalelor asociate, de lungime maximă, din care este format șirul dat sau mesajul multimea vida, în cazul în care acestea nu conțin niciun element comun.

**Exemplu:** Dacă fișierul conține numerele 2 3 4 1 2 3 4 5 6 2 3 4 5 0 1 2 3 atunci pe ecran vor fi afișate valorile 2 3 deoarece șirul este format din 4 intervale asociate 2 4, 1 6, 2 5 și 0 3 a căror intersecție este intervalul 2 3. Dacă fișierul conține numerele 3 4 5 4 5 6 8 atunci pe ecran se va afișa multimea vida.

- a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de execuție și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. (2 puncte)
- b) Scrieți programul C/C++ corespunzător metodei descrise la punctul a). (8 puncte)

## Teza 6

(20 de puncte)

## SUBIECTUL I

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

- Variabilele  $x$ ,  $y$  și  $z$  sunt de tip întreg. Care dintre următoarele expresii C/C++ are valoarea 1 dacă și numai dacă cel mult două dintre variabilele precizate au valori egale?
  - $!(x==y \ \&\& \ x==z) \ || \ y!=z$
  - $x==y \ || \ x==z \ || \ y==z$
  - $(x-y) * (x-z) * (y-z) == 0$
  - $x==y==z$
- Indicați ce condiție verifică variabila  $a$  de tip real din expresia C++  $\text{sqrt}(a)*10+\text{pow}(a,2)<100$  astfel încât expresia să aibă valoarea 1.
  - $a > -5 \ \&\& \ a < 9$
  - $a > 3 \ \&\& \ a < 10$
  - $!(a < 0 \ || \ a >= 9)$
  - $!(a < 0 \ \&\& \ a >= 9)$
- Fiind date 3 numere naturale nenule  $a$ ,  $b$ ,  $k$  ( $a \leq b$ ), numărul de numere din intervalul  $[a, b]$  divizibile cu  $k$  este:
  - $b/k - a/k$
  - $b/k - a/k + a\%k$
  - $b/k - (a-1)\%k$
  - $b/k - a\%k$
- Se dau vectorii  $a = (1, 3, 5)$ ,  $b = (2, 4, 6)$ . Utilizând algoritmul de interclasare se obține vectorul  $c = (1, 2, 3, 4, 5, 6)$ . Care este numărul de comparații care s-au făcut pe parcursul algoritmului?
  - 5
  - 4
  - 6
  - 3
- Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

Ce se va afișa dacă se citesc numerele 5 3 2 22 9 21 81 63?

- 0
- 1
- 2
- 3

citește  $n$ ,  $k$ ,  $pk$  ( $n$ ,  $k$  numere naturale nenule)

$m \leftarrow 0$

pentru  $i \leftarrow 1, n$  execută

citește  $a$

$p \leftarrow 0$

cât timp  $a\%k=0$  atunci

$p \leftarrow p+1$

$a \leftarrow [a/k]$

dacă  $p=pk$  atunci

$m \leftarrow m+1$

scrie  $m$

## SUBIECTUL II

(40 de puncte)

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

- a) Ce se va afișa dacă  $a = 8$  și  $b = 12$ ? (6 puncte)
- b) Pentru  $a = 25$ , care este cea mai mare valoare pe care o poate primi  $b$  astfel încât să se afișeze doar „\*”? (6 puncte)
- c) Scrieți programul C/C++ corespunzător algoritmului dat. (10 puncte)
- d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască „cât timp...execută” cu alt tip de structură repetitivă. (6 puncte)

citește  $a, b$  ( $a, b$  numere naturale,  $a \leq b$ )  
 $y \leftarrow 0$

pentru  $x \leftarrow b, a, -1$  execută

$y \leftarrow x$

$k \leftarrow 0$

cât timp  $y > 0$  execută

$k \leftarrow k + 1 - 2 * (y \% 2)$

$y \leftarrow [y/2]$

■

dacă  $k = 0$  atunci

scrie  $x, ‘‘$

■

■

scrie ‘\*’

2. Se consideră tablourile unidimensionale **lungime**, **lățime** și **înălțime** formate din exact 100 de numere naturale fiecare ce memorează laturile a 100 de turnuri de formă paralelipiped. Știind că  $lungime[0]$ ,  $latime[0]$  și  $inaltime[0]$  reprezintă laturile primului turn scrieți o secvență de instrucțiuni C++ care determină și afișează pe ecran numărul de turnuri care au formă de cub. (6 puncte)

3. În secvența de instrucțiuni de mai jos, variabilele  $i$  și  $j$  sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. (6 puncte)

```
for (i=1; i<=6; i++) {
 for (j=1; j<=6; j++)
 }
```

```
2 3 1 2 3 1
3 6 2 3 6 2
1 2 0 1 2 0
2 3 1 2 3 1
3 6 2 3 6 2
1 2 0 1 2 0
```

(30 de puncte)

## SUBIECTUL III

1. Scrieți un program în limbaj pseudocod care citește de la tastatură două numere naturale nenule  $a$  și  $b$ , care au cel mult 9 cifre, determină și afișează cel mai mic număr de permutări circulare la dreapta, cu câte o poziție, ale cifrelor lui  $a$ , astfel încât să se obțină numărul  $b$ . Dacă nu e posibil, se va afișa mesajul **Imposibil**. (10 puncte)
- Exemplu:** Pentru  $a = 120362$  și  $b = 621203$  se va afișa 2, iar pentru  $a = 732$  și  $b = 237$  se va afișa **Imposibil**.

2. Se citește un număr natural  $k$  ( $1 \leq k \leq 30$ ), un tablou unidimensional  $v$ , având  $k$  numere naturale în ordine strict crescătoare și un număr natural  $n$  ( $1 \leq n \leq 99999$ ). Scrieți un program C/C++ care determină cel mai mic divizor prim al lui  $n$  și îl caută în vectorul  $v$ . Dacă divizorul prim va fi găsit, programul va afișa indicele elementului din vector egal cu el. În caz contrar, divizorul prim va fi adăugat în  $v$  pe poziția corespunzătoare astfel încât ordinea strict crescătoare să se păstreze și se va afișa vectorul  $v$ , obținut după inserare. În vectorul  $v$ , indicii elementelor se notează începând de la 1.
- Exemplu:** Dacă  $k=4$ ,  $v = (2, 5, 11, 17)$ ,  $n=27$  se va afișa  $v = (2, 3, 5, 11, 17)$ , iar dacă  $k=4$ ,  $v = (2, 5, 11, 17)$ ,  $n=121$ , atunci se va afișa 3. (10 puncte)

3. Un număr este **palindrom** dacă citind cifrele numărului de la dreapta la stânga și de la stânga la dreapta se obține același număr. De exemplu 12321, 2002 sunt palindroame, iar 12312 nu este palindrom.

Fișierul text `bac.txt` conține pe prima linie  $n$ , un număr natural nenul, mai mic decât  $10^5$ , iar pe a doua linie un șir de  $n$  numere din intervalul  $[1, 9]$ , separate prin câte un spațiu. Se cere afișarea pe ecran a celui mai mare palindrom care se poate forma cu toate cifrele date. Dacă nu se poate construi un palindrom cu toate cifrele lui  $n$ , se va afișa -1. Proiectați un algoritm eficient din punct de vedere al memoriei utilizate și al timpului de executare.

**Exemplu:** Dacă fișierul `bac.txt` conține numerele:

10

2 3 3 8 9 2 3 9 8 3 atunci pe ecran se afișează: 9833223389.

- a) Descrieți în limbaj natural algoritmul utilizat, justificând eficiența acestuia. (2 puncte)

- b) Scrieți programul C/C++ corespunzător algoritmului descris la punctul a). (8 puncte)

## SUBIECTUL I

(20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

- Știind că variabila reală  $x$  aparține reuniunii intervalelor  $[3, 7]$  și  $(20, 30)$  care dintre expresiile scrise mai jos, NU are valoarea 1?
  - $!((x < 3 \parallel x > 7) \&\& (x \geq 30 \parallel x \leq 20))$
  - $(x \leq 7 \&\& x \geq 3) \&\& (x > 20 \&\& x < 30)$
  - $(x \leq 7 \&\& x \geq 3) \parallel (x > 20 \&\& x < 30)$
  - $!(x < 3 \parallel x > 7) \parallel !(x \geq 30 \parallel x \leq 20)$
- Indicați cel mai mic și cel mai mare număr natural pe care îl poate memora variabila  $a$ , astfel încât expresia `int(sqrt(a+100)) == 8+floor(a/10)` să aibă valoarea 1.
  - 40 și 49
  - 40 și 68
  - 44 și 49
  - 44 și 68
- Ce valori au variabilele  $x$  și  $y$ , astfel încât prin interclasarea tablourilor  $(2, 5, 6, x, 30)$  și  $(1, y, 7, 20)$  să se obțină tabloul  $(1, 2, 2, 5, 6, 7, 10, 20, 30)$ ?
  - $x = 7, y = 2$
  - $x = 10, y = 7$
  - $x = 7, y = 10$
  - $x = 10, y = 2$
- Se consideră tabloul  $x = (128, 45, 602, 22, 7)$ . Pentru fiecare element din vector se calculează suma cifrelor și se rearanjează elementele din tabloul  $x$ , astfel încât să fie ordonat crescător, în funcție de suma cifrelor lor. Care este tabloul obținut după ordonare?
  - $(7, 22, 45, 128, 602)$
  - $(22, 7, 128, 45, 602)$
  - $(22, 7, 602, 45, 128)$
  - $(22, 45, 7, 128, 602)$

- Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

Ce se va afișa dacă se citesc numerele 7 22 8 21 15 7 28 3?

a) 12

b) 14

c) 16

d) 18

```

citește n (n număr natural nenul)
s ← 0; d ← 2
pentru i ← 1, n execută
 citește a
 dacă a % d = 0 atunci
 s ← s + d
 altfel
 s ← s + 1
 d ← d + 1
scrie s

```



## SUBIECTUL II

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

a) Scrieți ce se va afișa dacă se citesc, în această ordine, numerele 5, 3, 2334, 31, 1535, 330, 75. (6 puncte)

b) Scrieți un set de date de intrare care să determine afișarea valorii 1000. (6 puncte)

c) Scrieți programul C++ corespunzător algoritmului dat. (10 puncte)

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „pentru” cu alt tip de structură repetitivă. (6 puncte)

citește  $n, y$  ( $y$  număr natural,  $y < 10$ ,  $n$  număr natural nenul)

$e \leftarrow 0$

pentru  $i \leftarrow 1, n$  execută

citește  $x$  (număr natural nenul)

$cx \leftarrow x$

$nr \leftarrow 0$

cât timp  $x \neq 0$  execută

dacă  $x \% 10 = y$

atunci

$nr \leftarrow nr + 1$

$x \leftarrow [x/10]$

dacă  $nr \% 2 = 0$  și  $nr > 0$

atunci

$e \leftarrow e + cx$

scrie  $e$

2. Se consideră tablourile  $latura1$ ,  $latura2$  și  $latura3$  formate din cel mult 100 de numere naturale fiecare și un număr natural  $n$  ( $2 \leq n \leq 100$ ). Știind că  $latura1[0]$ ,  $latura2[0]$  și  $latura3[0]$  reprezintă laturile primului triunghi și că  $latura1[n-1]$ ,  $latura2[n-1]$  și  $latura3[n-1]$  reprezintă laturile ultimului dintre cele  $n$  triunghiuri date, scrieți o secvență de instrucțiuni C++ care determină și afișează pe ecran valoarea maximă și valoarea minimă dintre perimetrele celor  $n$  triunghiuri date. (6 puncte)

3. În secvența de instrucțiuni de mai jos variabilele  $i$  și  $j$  sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze caracterele de mai jos, în această ordine. (6 puncte)

```
for (i=0; i<6; i++) {
 for (j=0; j<6; j++)

 cout<<endl; }
```

```
A B C D E F
B A B B F B
C C A F C C
D D F A D D
E F E E A E
F F F F F A
```

1. Scrieți un program în limbaj pseudocod care citește de la tastatură un număr natural ( $1 \leq n \leq 100$ ) și afișează termenul cu numărul de ordine egal cu  $n$  din șirul următor: 1, 2, 3, 2, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 2, 3, 4, 5, 6, 7, 8, 9, 8, 7, 6, 5, 4, 3, 2, 1 etc.  
**Exemplu:** Dacă  $n = 10$  atunci se va afișa numărul 6. (10 puncte)
2. Scrieți un program C++ care citește de la tastatură un număr natural  $n$  ( $1 \leq n \leq 100$ ) și un tablou unidimensional  $x$  format din  $n$  numere întregi și afișează produsul maxim obținut din două elemente distincte ale tabloului.  
**Exemplu:** Pentru  $n = 6$  și tabloul  $x = (-60, 36, -8, -2, 10, 5)$ , se va afișa 480. (10 puncte)
3. Considerând un număr natural  $x$  ce conține cel puțin trei cifre, după ce se elimină prima și ultima cifră se obține un alt număr denumit **număr interior** al numărului  $x$ . Fișierul **bac.in** conține un șir având cel mult un milion de numere naturale care au cel puțin trei cifre și cel mult 5 cifre, separate prin câte un spațiu. Scrieți un program C++ care afișează pe ecran în ordine descrescătoare, separate prin spațiu, numerele interioare ale numerelor din fișier pentru care prima și ultima lor cifră sunt egale sau afișează mesajul Nu există, dacă fișierul nu conține niciun număr având prima și ultima cifră egale. Utilizați un algoritm eficient ca timp de execuție și spațiu de memorie utilizat.  
**Exemplu:** Dacă fișierul conține șirul de numere 1151 234 2322 212 514 23122 atunci se vor afișa numerele 312 32 15 1.
- a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. (2 puncte)
- b) Scrieți programul C/C++ corespunzător algoritmului descris. (8 puncte)



## Teza 8

(20 de puncte)

## SUBIECTUL I

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Stabiliți care dintre următoarele expresii C++ are valoarea 1, dacă și numai dacă numerele naturale nenule  $x$  și  $y$  au aceeași paritate (ambele sunt numere pare sau ambele sunt numere impare).

a.  $x + y \% 2 == 0$       b.  $(x + y) \% 2 == 0$       c.  $(x * y) \% 2 == 0$       d.  $(x * y) \% 2 == 1$

2. Variabilele  $a$ ,  $b$ , și  $c$  sunt de tip real. Indicați expresia C++ care reprezintă expresia aritmetică următoare:  $\frac{(b+c)^2}{2a^2}$ .

a)  $\text{pow}(b,2) + \text{pow}(c,2) / (2 * \text{pow}(a,2))$       c)  $\text{pow}(b+c,2) / (2 * \text{pow}(a,2))$   
b)  $\text{pow}(b,2) + \text{pow}(c,2) / 2 * \text{pow}(a,2)$       d)  $\text{pow}(b+c,2) / 2 * \text{pow}(a,2)$

3. Se consideră un tablou  $x$ , format din 10 elemente, numere naturale și variabilele  $i$  și  $s$ , de tip întreg. Completați punctele de suspensie din secvența de program C++ următoare, astfel încât să se afișeze suma elementelor tabloulului  $x$ , divizibile cu 2 și cu 7.

```
s=0;
for(i=0; i<10; i++)
 if(...)
 s=s+x[i];
cout<<s;
```

a)  $i \% 2 == 0 \ \&\& \ i \% 7 == 0$       c)  $x[i] \% 2 != 0 \ \&\& \ x[i] \% 7 != 0$   
b)  $x[i] \% 2 == 0 \ || \ x[i] \% 7 == 0$       d)  $x[i] \% 2 == 0 \ \&\& \ x[i] \% 7 == 0$

4. Se consideră tabloul  $a = (10, -2, 4, 3, 1)$ . Pentru a căuta valoarea  $x = 3$  în vectorul  $a$ , în mod eficient se vor aplica algoritmii următori:

a) interclasare      c) ordonare și căutare binară  
b) cautare binară      d) căutare secvențială

5. Se consideră o secvență de program, în care  $i$ ,  $j$ ,  $n$  și  $s$  sunt variabile întregi.

```
s=0; i=1;
do
{ if(i+s<=n)
 { s=s+i;
 for(j=1; j<=i; j++)
 cout<<j<<" ";
```

```

i++;
}
else
s=n;
}
while(s<n);

```

Ce valoare are numărul  $n$  astfel încât la execuția acestei secvențe de program șirul numerelor afișate să conțină numărul 1 de patru ori?

- a) 27                      b) 16                      c) 11                      d) 4

## SUBIECTUL II

(40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

- a) Scrieți valoarea care se va afișa pentru  $a = 3$  și  $b = 7$ . (6 puncte)
- b) Scrieți un set de valori pentru  $a$  și  $b$  astfel încât rezultatul afișat să fie  $k = 8$  și  $q = 4$ . (6 puncte)
- c) Scrieți programul C++ corespunzător algoritmului dat. (10 puncte)
- d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască prima structură „pentru” cu alt tip de structură repetitivă. (6 puncte)

citește  $a, b$  ( $a \leq b$ ,  $a, b$  numere naturale)

$k \leftarrow 0; q \leftarrow 0$

pentru  $x \leftarrow 1$ , a execută

pentru  $y \leftarrow 1$ , b execută

dacă  $y \% x = 0$

atunci

$k \leftarrow k + y$

altfel

$q \leftarrow q + y$

scrie  $k, ",", q$

2. Pentru evidența unor date despre 200 de orașe, se consideră două tablouri: **loc** ce memorează numărul de locuitori ai fiecărui oraș și **supr** ce memorează suprafața fiecărui oraș (exprimată în număr de  $\text{km}^2$ ). Orașele sunt notate cu numere naturale de la 1 la 200, denumite **coduri**. Astfel, orașul care are codul 1 are  $\text{loc}[1]$  locuitori și suprafața egală cu  $\text{supr}[1]$   $\text{km}^2$ , orașul 2 are datele  $\text{loc}[2]$  și  $\text{supr}[2]$  etc. Știind că densitatea populației reprezintă numărul de persoane care locuiesc pe un  $\text{km}^2$ , scrieți o secvență de instrucțiuni în limbajul C++ care să afișeze codurile tuturor orașelor care au o densitate mai mare de 15 locuitori/ $\text{km}^2$ . (6 puncte)

3. În secvența de instrucțiuni de mai jos variabilele  $i$  și  $j$  sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. (6 puncte)



```
for(i=0;i<5;i++){
 for(j=0;j<5;j++)
 {
 cout<<endl;}
}
```

```
6 7 8 9 10
7 8 9 10 11
8 9 10 11 12
9 10 11 12 13
10 11 12 13 14
```

### SUBIECTUL III

(30 de puncte)

1. Scrieți un program în limbaj pseudocod care citește de la tastatură un număr natural  $n$  ( $2 \leq n \leq 50$ ) și  $n$  numere naturale distincte, fiecare număr având cel mult 9 cifre și afișează numărul de perechi din șir, formate din numere introduse consecutiv, cu proprietatea că un număr este oglindit pentru celălalt număr.

**Exemplu:** Dacă  $n = 7$  și  $v = (41, 14, 324, 423, 6, 82, 28)$  atunci se va afișa valoarea 3, deoarece șirul de numere conține 3 perechi de numere ce verifică proprietatea dată: (41, 14); (324, 423); (82, 28). (10 puncte)

2. Se citesc de la tastatură un număr natural  $n$  ( $1 \leq n \leq 50$ ) și un tablou unidimensional  $v$  format din  $n$  numere naturale nenule. Scrieți un program C++ care afișează cel mai mic factor prim comun tuturor elementelor din tabloul  $v$  notat  $fp$  și puterea factorului  $fp$ , notată cu  $p$ , pentru care numărul  $fp^p$  este divizor pentru fiecare element din tabloul  $v$ , cu  $p > 0$ . Dacă elementele tabloului  $v$  nu au un factor prim comun atunci  $fp$  are valoarea  $-1$  și  $p$  are valoarea 0.

**Exemplu:** Pentru  $n = 4$  și tabloul  $v = (60, 36, 18, 24)$ , se va afișa  $fp = 2, p = 1$ , iar pentru  $n = 4$  și tabloul  $v = (22, 60, 21, 25)$ , se va afișa  $fp = -1$  și  $p = 0$ . (10 puncte)

3. Scrieți un program C++, eficient ca timp de execuție și spațiu de memorie utilizat, care afișează în fișierul **bac.txt**, în ordine strict crescătoare, pe câte o linie, toate palindroamele formate din 6 cifre impare. Primele 3 numere afișate în fișier sunt: 111111, 113311, 115511.

a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. (2 puncte)

b) Scrieți programul C/C++ corespunzător algoritmului descris. (8 puncte)

## Teza 9

### SUBIECTUL I

(20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Stabiliți ce valori au variabilele întregi  $n$  și  $x$ , astfel încât la execuția secvenței de program următoare să se afișeze patru caractere  $*$  urmate de șase caractere  $\#$ .

```
int i=1,n,x;
while(i*i<=n)
{cout<<'*'; i++;}
while(i<=n+x)
{cout<<'#'; i++;}
```

- a)  $n = 18, x = 2$       b)  $n = 5, x = 6$       c)  $n = 16, x = -6$       d)  $n = 24, x = 5$

2. Completați punctele din suspensie din secvența de program următoare, astfel încât, prin execuția sa să determine prin variabila `divizor_maxim` cel mai mare divizor propriu al unui număr natural  $n$ , nenul.

```
int n,d,divizor_maxim=0;
for(d=...;d<=...;d++)
 if(n%d==0)
 if(...)
 divizor_maxim=d;
```

- a)  $1, n/2, d > \text{divizor\_maxim}$   
b)  $2, n, d > \text{divizor\_maxim}$

- c)  $2, n/2, d > \text{divizor\_maxim}$   
d)  $2, \text{sqrt}(n), d < \text{divizor\_maxim}$

3. Se consideră tabloul  $x = (23, 16, 10, 25, 17)$ . Pentru fiecare element din vector se calculează numărul de divizori și se rearanjează elementele din tabloul  $x$ , astfel încât să fie dispuse în ordine descrescătoare în funcție de numărul lor de divizori. Care este tabloul obținut după ordonare?

- a)  $(25, 23, 17, 16, 10)$   
b)  $(16, 10, 25, 23, 17)$

- c)  $(17, 23, 25, 10, 16)$   
d)  $(10, 16, 17, 23, 25)$

4. Prin aplicarea algoritmului de interclasare a două tablouri  $a$  și  $b$  se obține tabloul  $t = (2, 2, 4, 5, 7, 10, 25, 40)$ . Care sunt tablourile  $a$  și  $b$  pentru care s-a obținut tabloul  $t$ ?

- a)  $a = (2, 4, 7, 10), b = (2, 5, 25, 40)$   
b)  $a = (2, 4, 5, 10, 25), b = (4, 7, 40)$   
c)  $a = (4, 7, 25), b = (2, 5, 10, 17, 40)$   
d)  $a = (2, 4, 25, 10), b = (2, 7, 5, 40)$

**BACALAUREAT INFORMATICĂ. LIMBAJUL C++**

5. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și cu  $[x]$  partea întreagă a numărului real  $x$ .  
Ce se va afișa dacă se citește numărul 35?

- a) 35                      b) 5                      c) 7                      d) 1

```

citește a (a număr natural nenul)
m ← 0
pentru d ← a, 2, -1 execută
 dacă a%d=0
 dacă d%2=1
 atunci
 m ← d
 altfel
 m ← 2*d
scrie m

```

**SUBIECTUL II**

(40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

- a) Scrieți ce se va afișa dacă se citesc numerele 12, 36, 3. (6 puncte)  
b) Scrieți un set de date de intrare, cu proprietatea  $b - a = 2$ , pentru care se afișează trei numere. (6 puncte)  
c) Scrieți programul C++ corespunzător algoritmului dat. (10 puncte)  
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „pentru” cu alt tip de structură repetitivă. (6 puncte)

```

citește a, b, k (numere naturale nenule, a ≤ b, k > 1)
pentru x ← a, b execută
 y ← 0
 t ← x
 cât timp t ≠ 0 execută
 y ← y*10 + t%10
 t ← [t/10]
 dacă y%k=0
 atunci
 scrie x, " "

```

2. Se consideră declarațiile de mai jos, prin care se memorează date despre 10 filme difuzate la un cinematograful pentru o singură zi.

```
int suma[11], suma_totala, suma_prec[11];
```

Pentru fiecare film se memorează: **suma** ce reprezintă suma totală încasată din vânzarea билетelor la data difuzării, și **suma\_prec** ce reprezintă suma totală încasată din vânzarea

biletelor în ziua precedentă. Codul fiecărui film este egal cu numărul său de ordine din tabloul **suma**. Astfel **suma[1]** reprezintă suma încasată pentru filmul care are codul egal cu 1 și pentru care s-a încasat **suma\_prec[1]** în ziua precedentă etc. Scrieți o secvență de instrucțiuni C++ care să determine și să afișeze pe ecran codurile filmelor care au avut încasări mai mari față de ziua precedentă și suma totală încasată de cinematograful, pentru toate filmele difuzate în ziua respectivă prin variabila **suma\_totala**. (6 puncte)

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. (6 puncte)

```
for(i=0;i<5;i++){
 for(j=0;j<5;j++)
 {
 cout<<endl;}
```

```
0 1 2 3 4
2 3 4 5 6
0 1 2 3 4
6 7 8 9 10
0 1 2 3 4
```

(30 de puncte)

### SUBIECTUL III

1. Scrieți un program în limbaj pseudocod care citește un număr natural **n**, având cel mult 8 cifre și afișează două numere naturale **m** și **t**, unde **m** reprezintă cel mai apropiat termen din șirul lui Fibonacci mai mic decât **n** și **t** reprezintă cel mai apropiat termen din șirul lui Fibonacci, mai mare decât **n**. (10 puncte)

Exemplu: Dacă **n = 30** se afișează: **m = 21, t = 34**.

2. Se consideră un număr natural **n** ( $2 \leq n \leq 50$ ) și un tablou unidimensional **a**, format din **n** numere naturale nenule și distincte, fiecare număr având cel mult 9 cifre. Scrieți un program C++ care determină și afișează numărul de perechi distincte formate din elemente din tabloul **a**, prime între ele. Două numere naturale sunt prime între ele dacă au un singur divizor comun: numărul 1. (10 puncte)

Exemplu: Dacă **n = 6** și **a = (42, 3, 14, 43, 8, 6)** atunci se va obține valoarea 7, deoarece vectorul are 7 perechi de numere ce verifică proprietatea dată: (42, 43); (3, 14); (3, 43); (3, 8); (14, 43); (43, 8); (43, 6).

3. Fișierul **bac.txt** conține pe prima linie un număr natural **n**, care este multiplu de 7 ( $7 \leq n \leq 70000$ ) și pe a doua linie un șir de **n** numere întregi nenule, având cel mult trei cifre. Șirul de numere se împarte în secvențe de câte 7 numere denumite **benzi** și pentru fiecare bandă se determină cele mai mici două numere distincte **min1** și **min2**. Scrieți un program



BACALAUREAT INFORMATICĂ. LIMBAJUL C++

C++, eficient ca timp de execuție și spațiu de memorie utilizat, care afișează pe ecran, pentru fiecare dintre benzi, în ordine crescătoare cele două valori **min1** și **min2** sau afișează numărul 0, dacă banda nu conține două valori distincte **min1** și **min2**.

**Exemplu:** Pentru  $n = 21$ , șirul de numere -3, 10, 1, 2, -5, 6, -5, 1, 1, 1, 1, 1, 1, 1, 10, 9, 1, 2, 3, 6, 7 se va afișa -5, -3, 0, 1, 2.

- a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. (2 puncte)
- b) Scrieți programul C/C++ corespunzător algoritmului descris. (8 puncte)

## Teza 10

### BIECTUL I

(20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

Variabilele  $a, b$  au valori numere naturale nenule,  $a \leq b$ . Care dintre următoarele instrucțiuni C/C++ următoare atribuie variabilei întregi  $c$  numărul de valori pare din intervalul  $[a, b]$ ?

- a)  $c = (b - a + 1) / 2;$
- b)  $c = a / 2 - b / 2;$
- c)  $c = b / 2 - (a - 1) / 2$
- d)  $c = (a + b) / 2;$

Care dintre următoarele expresii are valoarea 1 pentru orice număr natural  $x$  din intervalul  $[a, b]$ ,  $a \leq b$ .

- a)  $(x - a) * (x - b) > 0$
- b)  $(x - a) * (b - x) < 0$
- c)  $(x - a) * (x - b) \leq 0$
- d)  $(a - x) * (b - x) > 0$

Se consideră secvența de program C++ următoare.

```
int n, m, i, j, k;
for (i = 1; i <= n; i++) {
 for (j = 1; j <= m; j++)
 if ((i + j) % 2 == 0)
 cout << i + j;
 for (k = 1; k <= i; k++)
 cout << "***";
}
```

Ce valoare au variabilele  $n$  și  $m$  astfel încât prin execuția programului ce conține această secvență să se afișeze  $246*468**468***6810****$ .

- a)  $n = 3, m = 5$
- b)  $n = 4, m = 4$
- c)  $n = 6, m = 4$
- d)  $n = 4, m = 6$

4. Variabilele  $x, y, a, b$  memorează numere naturale. Care dintre următoarele expresii este adevărată dacă și numai dacă  $[x, y] \subset [a, b]$ .

- a)  $a \leq x \mid b \leq y$
- b)  $a \leq x \&\& y \leq b$
- c)  $! (a \leq x \&\& y \leq b)$
- d)  $x \leq a \&\& b \leq y$

BACALAUREAT INFORMATICĂ. LIMBAJUL C++

5. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

Ce valoare se va afișa pentru  $x = 140$  și  $y = 15$ ?

- a) 1 125      b) 8 1      c) 9 5      d) 10 4

```

citește x, y
(x, y numere naturale)
r ← x
cât timp y ≤ r execută
 r ← r - y
 ■
k ← (x - r) / y
scrie k, " ", r

```

SUBIECTUL II

(40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu  $x \% y$  restul împărțirii numerelor întregi  $x$  și  $y$  și cu  $[x]$  partea întreagă a numărului real  $x$ .

- a) Scrieți valoarea care se va afișa pentru  $n = 23100$ .  
(6 puncte)
- b) Scrieți cel mai mic și cel mai mare număr de două cifre care poate fi citit pentru  $n$  astfel încât algoritmul să afișeze valoarea 1.  
(6 puncte)
- c) Scrieți programul C/C++ corespunzător algoritmului dat.  
(10 puncte)
- d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura repetitivă „cât timp... execută” cu o structură repetitivă cu condiție finală.  
(6 puncte)

```

citește n (n număr natural, n > 1)
p ← 1; m ← 0; x ← 2
cât timp n ≥ x execută
 dacă n % x = 0 atunci
 m ← m + 1; n ← [n / x]
 altfel
 dacă m % 2 ≠ 0 atunci
 p ← p * x
 m ← 0; x ← x + 1
 ■
 ■
 dacă m % 2 ≠ 0 atunci
 p ← p * x
 ■
scrie p

```

2. Se consideră variabilele  $a$ ,  $b$  și  $c$  ce memorează simultan lungimile laturilor unui triunghi (numere reale pozitive). Fără a utiliza variabile suplimentare, scrieți o expresie C++ care are valoarea 1 dacă și numai dacă trunghiul ale cărui laturi sunt memorate în variabilele  $a$ ,  $b$  și  $c$  este un triunghi isoscel, fără a fi triunghi echilateral.  
(6 puncte)
3. Se consideră secvența următoare de program C++ în care se utilizează două tablouri unidimensionale  $a$  și  $b$ , formate din numere naturale. Tabloul  $a$  conține  $n$  numere pare în ordine crescătoare și tabloul  $b$  conține  $m$  numere impare în ordine crescătoare. Determinați două tablouri  $a$  și  $b$ , care să conțină un număr minim de elemente, astfel

Încât secvența dată să afișeze 5 valori în ordine crescătoare, separate prin spațiu, alese alternativ din tablourile a și b. (6 puncte)

```
int i=0, j=0;
while(i<n && j<m)
 if(a[i]<b[j])
 {cout<<a[i]<<" "; i++;}
 else
 {cout<<b[j]<<" "; j++;}
```

### BIECTUL III

(30 de puncte)

Se citește un număr natural  $n$  și un șir de  $n$  numere naturale cu cel mult trei cifre fiecare. Scrieți un algoritm în pseudocod care afișează, în ordinea citirii, numerele din șir cu proprietatea că reprezentarea lor în baza 2 conține doar cifre egale cu 1. (10 puncte)

**Exemplu:** Pentru  $n = 5$  și numerele 7 18 15 22 31 se vor afișa 7 15 31.

Un tablou unidimensional se numește **k-palindrom** dacă după efectuarea a  $k$  permutări circulare cu o poziție spre stânga acesta devine palindrom (considerăm că un vector este palindrom dacă vectorul parcurs de la stânga la dreapta coincide cu vectorul parcurs de la dreapta la stânga). Se citește de la tastatură un număr natural  $n$  ( $n \leq 100$ ), un tablou unidimensional  $v$  format din  $n$  numere întregi și un număr natural  $k$  ( $1 \leq k < n$ ). Scrieți un program C++ care afișează mesajul DA dacă vectorul este k-palindrom sau mesajul NU în caz contrar. (10 puncte)

**Exemplu:** Dacă  $n = 5$ ,  $v = (2, 2, 4, 5, 4)$ ,  $k=1$  se va afișa DA.

Fișierul text **BAC.TXT** conține un șir  $S$  cu cel mult 100 000 de numere naturale din intervalul  $[2, 10^9]$ . Pentru fiecare valoare  $x$  din șir se determină numărul de cifre egale cu zero de la sfârșitul lui  $x!$  (unde  $x! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot x$ ). Scrieți un program C/C++ care citește șirul  $S$ , obține și afișează pe ecran, cu spațiu între ele, numărul de valori  $x$  din șir care au un număr maxim de cifre egale cu zero la sfârșitul lui  $x!$  și care este această valoare maximă. (2 puncte)

**Exemplu:** Dacă fișierul conține numerele 102 12 50 100 atunci pe ecran vor fi afișate valorile 2 24. (8 puncte)

- Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. (2 puncte)
- Scrieți programul C/C++ corespunzător metodei descrise la a). (8 puncte)

## PARTEA a III-a

# SOLUȚII

## Indicații de rezolvare, răspunsuri, rezolvări complete

### BREVIAR

#### CAPITOLUL 1. Elemente de bază ale limbajului C++

$!(a \% 2 == 0 \parallel b \% 2 == 0) \&\& !(a * b < 0) \Leftrightarrow (a \% 2 != 0 \&\& b \% 2 != 0) \&\& (a * b >= 0)$

$a >= n \parallel b <= n \&\& (n \% 3 == 0 \&\& n \% 673 == 0)$

$x / 10 \% 10 + x \% 10 * 10 == x / 100 / 10$

-4

$(5 * 4 \% 7 - 19 / (7 - 4 \% 5)) / 2$  este 0, deoarece  $(20 \% 7 - 19 / (7 - 4)) / 2 = (6 - 19 / 3) / 2 = (6 - 6) / 3 = 0$

12, deoarece sunt 2 atribuiri  $n = 1023$ ,  $k = 2$ , și alte 8 (câte două la fiecare iterație a instrucțiunii while)

Ordinea de execuție a celor 3 operații poate fi: 3, 1, 2 sau 2, 1, 3

unde numerotăm operațiile astfel: 1)  $x = y - x$  2)  $y = x + y$  3)  $y = y - x$

Instrucțiunea C/C++ corespunzătoare următoarei operații de atribuire:  $a \leftarrow \frac{(x + y^2) \cdot (y - 1)}{x^2 \cdot y}$

este  $a = (x + y * y) * (y - 1) / (x * x * y)$

Expresiile C/C++ de mai jos au valoarea 1 dacă și numai dacă numărul memorat în variabila  $z$  este fie multiplu de  $x$  și de  $y$ , fie este divizor al lui  $x$  și al lui  $y$ .

a)  $(z \% x + z \% y == 0) \parallel !(x \% z \parallel y \% z)$

#### CAPITOLUL 2. Algoritmi elementari

a) Răspuns: 32. Se obține  $a = 16 = 2^4$ ,  $i = 2$ ,  $i * a = 32$

b) Cea mai mică valoare de valoare de 3 cifre care se poate citi pentru  $a = 103$ , astfel încât valoarea afișată de algoritmul alăturat este egală cu  $162 = 2 * 81$  ( $3^4 = 81$  și  $i = 2$ ) și este maxim.

c) Algoritm echivalent:

```
citește a
i ← 0
a ← a % 10
dacă a > 1 și a < 10
 atunci
 repetă
 i ← i + 1
 a ← a * a
 până când a ≥ 10
scrie i * a
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int main () {
 int a, i;
 cin >> a; i = 0;
 a = a % 10;
 while (a > 1 && a < 10) {
 i++;
 a = a * a;
 }
 cout << i * a;
 return 0; }
```

2. a) Răspuns: 32138. Algoritmul construiește și afișează un număr de  $n$  cifre, format din prima cifră a fiecărui număr din șirul dat, în ordinea citirii.

b) Pentru ca algoritmul să afișeze 9999, se va citi un șir de 4 numere, fiecare număr poate avea o cifră egală cu 9 sau poate avea 2 cifre, prima fiind egală cu 9.

De exemplu: se pot citi 4 93 9 96 98 și se va afișa 9999.



BACALAUREAT INFORMATICĂ. LIMBAJUL C++

c) Algoritm echivalent

```

citește n
a ← 0
pentru i ← 1, n execută
 citește x
 p ← 1
 cât timp p ≤ [x/10] execută
 p = p * 10;
 a ← a * 10 + [x/p]
scrie a

```

d) Program C/C++

```

#include <iostream>
using namespace std;
int main ()
{
 int a,i;
 cin >> a; i = 0;
 a = a % 10;
 while (a > 1 && a < 10)
 {
 i++;
 a = a * a;
 }
 cout << i * a;
 return 0;
}

```

3. a) Răspuns: 13 23 17 19. Algoritmul afișează perechile de numere prime între ele de forma (p, u), dintre numerele din intervalul [a, b], care sunt simetrice față de mijlocul intervalului.

- b) (4,9), (5,8), (6,7).

c) Algoritm echivalent

```

citește a, b (numere naturale nenule, a < b)
p ← a; u ← b
cât timp p < u execută
 x ← p; y ← u
 cât timp y ≠ 0 execută
 z ← x % y
 x ← y
 y ← z
 dacă x = 1 atunci
 scrie p, ', ', u, ', '
 p ← p + 1; u ← u - 1

```

d) Program C/C++

```

#include <iostream>
using namespace std;
int main() {
 int a, b, p, u, x, y, z;
 cin >> a >> b;
 p = a; u = b;
 while (p < u) {
 x = p; y = u;
 do {
 z = x % y;
 x = y;
 y = z;
 } while (y != 0);
 if (x == 1)
 cout << p << " " << u << " ";
 p = p + 1; u = u - 1;
 }
 return 0;
}

```

4. a) Răspuns: 23. Algoritmul realizează conversia numărului a, din baza b în baza 10 și afișează rezultatul conversiei efectuate.

- b) a = 1057 și b = 8.

c) Algoritm echivalent

citește a, b (numere naturale nenule,  $2 \leq b \leq 9$ )

```
p ← 1
x ← 0
cât timp a > 0 execută
 x ← x + (a % 10) * p
 p ← p * b
 a ← [a/10]
```

scrie x

d) Program C/C++

```
#include <iostream>
using namespace std;
int main() {
 int a, b, m, x, p, i;
 cin >> a >> b;
 m = 0; x = 0;
 while (a > 0) {
 p = a % 10;
 for (i = 1; i <= m; i++)
 p = p * b;
 x = x + p;
 a = a / 10;
 m++;
 }
 cout << x;
 return 0;}
```

a) Răspuns: 3

b) Valori pentru b: 1 5 25

c) Algoritm echivalent

citește n, a (număr natural nenul)

```
x ← 0
pentru i ← 1, n execută
 citește b (numere naturale nenule, a ≤ b)
 p ← 1
 dacă p < b atunci
 repetă
 p ← p * a
 până când p ≥ b
 x ← x + [b/p]
```

scrie x

d) Program C/C++

```
#include <iostream>
using namespace std;
int n, x, i, a, b, p;
int main() {
 cin >> n >> a;
 x = 0;
 for (i = 1; i <= n; i++) {
 cin >> b;
 p = 1;
 while (p < b)
 p = p * a;
 x = x + b / p;
 }
 cout << x;
 return 0;}
```

6. a) Răspuns: 9 4 2  
 b) În intervalul  $[a, b]$  trebuie să existe doar un număr pătrat perfect de număr prim.  
 De exemplu:  $a = 40$  și  $b = 50$ .

**BACALAUREAT INFORMATICA. LIMBAJUL C++**

**c) Algoritm echivalent**

```

citește x, y (numere naturale nenule, $x \leq y$)
 $k \leftarrow 0$
pentru $y \leftarrow y, x, -1$ execută
 $d \leftarrow 2$
 cât timp $d * d \leq y$ și $y \% d \neq 0$ execută
 $d \leftarrow d + 1$
 dacă $d * d = y$ atunci
 scrie y, ' '
 $k \leftarrow k + 1$
scrie k

```

**d) Program C/C++:**

```

#include <iostream>
using namespace std;
int x, y, k, d;
int main() {
 cin >> x >> y;
 k = 0;
 while (y >= x) {
 d = 2;
 while (d * d < y && y % d != 0)
 d++;
 if (d * d == y) {
 cout << y << " ";
 k++;
 }
 y = y - 1;
 }
 cout << k;
 return 0;
}

```

7. a) Răspuns: 89 21 8 2

b) Numărul n trebuie să fie cel mai mic număr de trei cifre ce este termen în șirul Fibonacci, **n=144**.

**c) Algoritm echivalent**

```

citește n (număr natural nenul, $2 \leq n$)
 $k \leftarrow 0$
cât timp $n > 0$ execută
 $a \leftarrow 1; b \leftarrow 1$
 dacă $b \leq n$ atunci
 repetă
 $b \leftarrow a + b$
 $p \leftarrow p * a$
 până când $b \geq n$
 scrie a, ' '
 $n \leftarrow n - a$
scrie k

```

**d) Program C/C++:**

```

#include <iostream>
using namespace std;
int n, k, a, b;
int main() {
 cin >> n;
 k = 0;
 while (n > 0) {
 a = 1; b = 0;
 while (b <= n) {
 b = a + b;
 a = b - a;
 }
 cout << a << " ";
 n = n - a;
 }
 return 0;
}

```

Răspuns: 0

Numărul  $n$  trebuie să fie palindrom de cel mult două cifre. Se va afișa 19.

Algoritm echivalent

```
citește n (număr natural)
p ← 1; x ← 0;
dacă p < [n/p] atunci
 repetă
 x ← x*10 + [n/p] % 10
 p ← p*10
 până când p ≥ [n/p]
dacă x = [n/p] sau x = [n/(p/10)] sau n < 10
 atunci scrie 1
 altfel scrie 0
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int n, p, x, y, a, b;
int main() {
 cin >> n; p = 1; x = 0;
 while (p < n / p) {
 x = x * 10 + n / p % 10;
 p = p * 10;
 }
 if (x == n / p || x == n / (p / 10) || n < 10) cout << 1;
 else cout << 0;
 return 0; }
```

a) Răspuns: 42

b) Numărul  $n$  trebuie să fie pătrat perfect de cel puțin trei cifre, de exemplu 100 sau 121.

c) Algoritm echivalent

```
citește n (număr natural nenul, n > 1)
x ← 2; p ← 1
cât timp x ≤ n execută
 k ← 0;
 dacă n % x = 0 atunci
 repetă
 n ← [n/x]
 k ← k + 1
 până când b ≥ n
 dacă k % 2 ≠ 0 atunci
 p ← p * x
 x ← x + 1
scrie p
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int n, p, k, x;
int main() {
 cin >> n;
 x = 2; p = 1;
 while (x <= n) {
 k = 0;
 while (n % x == 0) {
 n = n / x;
 k++;
 }
 if (k % 2 != 0)
 p = p * x;
 x++;
 }
 cout << p;
 return 0; }
```

**BACALAUREAT INFORMATICĂ. LIMBAJUL C++**

**CAPITOLUL 3. Fișiere text**

1. Citim datele din fișier și aplicăm algoritmul de determinare a cmmdc și cmmmc.

```
#include <fstream>
using namespace std;
ifstream fin("numere.in");
ofstream fout("numere.out");
int n,i,a,b,ca,cb,r;
int main()
{
 fin>>n;
 for(i=1; i<=n; i++)
 {
 fin>>a>>b;
 ca=a;cb=b;
 while(b)
 {
 r=a%b;a=b;b=r;
 }
 fout<<a<<' ,<<ca*cb/a<<endl;
 }
 return 0;
}
```

2. Citim succesiv valorile din fișier și le comparăm cu cele 3 variabile m1, m2, m3.

```
#include <fstream>
#include <iostream>
using namespace std;
ifstream fin("numere.in");
int x,m1,m2,m3;
int main()
{
 m1=m2=m3=1000000000;
 while(fin>>x)
 {
 if(x%3==0)
 {
 if(x<m1)
 m3=m2,m2=m1,m1=x;
 else if(x<m2)
 m3=m2,m2=x;
 else if(x<m3)m3=x;
 }
 }
 cout<<m1<<' ,<<m2<<' ,<<m3;
 return 0;
}
```

3. La fiecare pas prelucrăm două valori aflate pe poziții consecutive în fișier.

```
#include <fstream>
#include <iostream>
using namespace std;
ifstream fin("numere.in");
int x,y;
int main()
{
 fin>>x;cout<<x<<' ,;
 while(fin>>y)
 {
 if(y!=x)cout<<y<<' ,;
 x=y;
 }
 return 0;
}
```

4. Aplicăm algoritmul de determinare a lungimii maxime a unei secvențe în care elementele au o anumită proprietate, în acest caz divizibilitatea cu 10.

```
#include <fstream>
#include <iostream>
using namespace std;
ifstream fin("numere.in");
int x,lg,lgmax;
int main()
{
 while(fin>>x)
 {
 if(x%10==0)lg++;
 else
 {
 if(lg>lgmax)
 lgmax=lg;
 lg=0;
 }
 if(lg>lgmax)lgmax=lg;
 cout<<lgmax;
 return 0;
 }
}
```

5. Citim succesiv valorile din fișier, fiecare valoare o împărțim la 2 atât timp cât este divizibilă cu 2, dacă în final s-a obținut valoarea 1, afișăm valoarea inițială.

```
#include <fstream>
#include <iostream>
using namespace std;
ifstream fin("numere.in");
int x,cx;
```

```

main()
while(fin>>x)
{
 cx=x;
 while(x%2==0)
 x=x/2;
 if(x==1) cout<<cx<<' ';
}
return 0;

```

Vom determina cea mai mică valoare cu toate cifrele egale mai mare decât a. Trecem succesiv prin toate valorile cu toate cifrele egale mai mici decât b.

```

#include <fstream>
#include <iostream>
#include <cmath>
using namespace std;
ofstream fout("bac.out");
int a,b,nra,na,i,p;
int main()
{
 cin>>a>>b;
 nra=log10(a)+1;
 na=0;
 for(i=1;i<=nra;i++) na=na*10+1;
 for(i=1;i<=9;i++)
 if(na*i>a) break;
 while(na*i<=b)
 {
 fout<<na*i<<' ';
 if(i==9) na=na*10+1, i=1;
 else i++;
 }
 return 0;
}

```

7. Parcurgem intervalul [a, b] și simulăm cerințele problemei.

```

#include <fstream>
#include <iostream>
using namespace std;
ofstream fout("bac.out");
int a,b,i,ci,si;
int main()
{
 cin>>a>>b;
 for(i=a;i<=b;i++)
 {
 ci=i; si=0;
 while(ci)

```

```

{
 si=si+ci%10;
 ci=ci/10;
}
if(i%si==i/si)
 fout<<i<<' ';
}
return 0;
}

```

8. Prefixele se obțin prin împărțiri succesive la 10.

```

#include <fstream>
#include <iostream>
using namespace std;
ofstream fout("bac.out");
int x;
int main()
{
 cin>>x;
 while(x)
 {
 fout<<x<<endl;
 x=x/10;
 }
 return 0;
}

```

9. Parcurgem numerele din fișier și determinăm numărul de ordine, până când găsim un număr divizibil cu 5 memorăm numărul și numărul de ordine al său.

```

#include <fstream>
#include <iostream>
using namespace std;
ifstream fin("numere.in");
int x,u,i,nr;
int main()
{

```

```

 while(fin>>x)
 {
 i++;
 if(x%5==0)
 nr=i, u=x;
 }
 cout<<u<<' '<<nr;
 return 0;
}

```

**BACALAUREAT INFORMATICA. LIMBAJUL C++**

**CAPITOLUL 4. Tablouri unidimensionale (vectori)**

```
1. #include <iostream>
#include <fstream>
using namespace std;
ifstream fin("numere.in");
int n,v[10000],i,aux,k;
int main()
{
 fin>>n;
 for(i=0;i<n;i++)
 fin>>v[i];
 k=n/2;
 aux=v[0];
 for(i=1;i<k;i++)v[i-1]=v[i];
 v[k-1]=aux;
 aux=v[n-1];
 for(i=n-2;i>=k;i--)
 v[i+1]=v[i];
 v[k]=aux;
 for(i=0;i<n;i++)
 cout<<v[i]<<' ';
 return 0;
}
```

```
2. #include <iostream>
using namespace std;
int n,vf[10],i,a,b,m;
int main()
{
 cin>>n;
 do
 {
 vf[n%10]++;
 n=n/10;
 }
 while(n);
 for(int i=9; i>=0; i--)
 if(vf[i])
 {
 for(int j=1; j<=vf[i];
 j++)a=a*10+i;
 b=b*10+i;
 }
 cout<<a<<endl<<b;
 return 0;
}
```

```
3. #include <fstream>
using namespace std;
ifstream fin("numere.in");
ofstream fout("numere.out");
int n,vf[100],i,x;
int main()
{
 while(fin>>x)vf[x]++;
```

```
for(i=0; i<100; i++)
 while(vf[i])
 {
 fout<<i<<' ';
 vf[i]--;
 }
 return 0;
}
```

```
4. #include <fstream>
using namespace std;
ifstream fin("numere.in");
ofstream fout("numere.out");
int n,vc[100],i,x;
int main()
{
 while(fin>>x)vc[x]=1;
 for(i=0; i<100; i++)
 if(vc[i])
 {
 fout<<i<<' ';
 }
 return 0;
}
```

```
5. #include <fstream>
using namespace std;
ifstream fin("numere.in");
ofstream fout("numere.out");
int n,p[100],u[100],i,x,dmax;
int main()
{
 while(fin>>x)
 {
 i++;
 if(p[x]==0)p[x]=i;
 u[x]=i;
 }
 for(i=0; i<100; i++)
 if(p[i])
 fout<<i<<'
 '<<u[i]-p[i]<<endl;
 return 0;
}
```

```
6. #include <fstream>
#include <iostream>
using namespace std;
ifstream fin("numere.in");
int n,v[1000],i,j;
int main()
{
 fin>>n;
 for(i=0;i<n;i++)
 fin>>v[i];
 for(i=0; i<n; i++)
```

```

if(v[i]%2==0)
{
 for(j=n-1; j>i; j--)
 v[j+1]=v[j];
 v[i+1]=2019;
 n++;i++;
}
for(i=0;i<n;i++)
 cout<<v[i]<<' ';
return 0;

```

```

#include <fstream>
#include<iostream>
using namespace std;
ifstream fin("numere.in");
int n,v[1000],i,j;
int main()
{
 fin>>n;
 for(i=0;i<n;i++)
 fin>>v[i];
 for(i=1; i<n; i++)
 if(v[i]==v[i-1])
 {
 for(j=i; j<n; j++)
 v[j]=v[j+1];
 n--;i--;
 }
 for(i=0;i<n;i++)
 cout<<v[i]<<' ';
 return 0;
}

```

```

8. #include <fstream>
#include<iostream>
using namespace std;
ifstream fin("bac.txt");
int n,v[500],x,i,j,ok,ls,ld,nr,mij;
int main()
{
 fin>>n;
 for(i=0; i<n; i++)
 fin>>v[i];
 fin>>x;
 //aplicam algoritmul de cautare
 binara
 ls=0;
 ld=n-1;
 ok=0;
 while(ls<=ld)
 {
 mij=(ls+ld)/2;

```

```

if(v[mij]<=x)
{
 ok=1;
 nr=v[mij];
 ls=mij+1;
}
else ld=mij-1;
}
if(ok)cout<<nr;
else cout<<"nu exista";
return 0;
}

```

```

9. #include <fstream>
#include<iostream>
using namespace std;
ifstream fin("bac.txt");
int n,a[500],b[500],i,j,m,k;
int main()
{
 cin>>m>>n;
 for(i=0; i<m; i++)
 cin>>a[i];
 for(j=0; j<n; j++)
 cin>>b[j];
 //aplicam algoritmul de
 interclasare
 i=m-1;
 j=n-1;
 while(i>=0&&j>=0)
 {
 if(a[i]>b[j])
 {
 if(a[i]%2!=0)
 {
 k++;
 cout<<a[i]<<' ';
 }
 i--;
 }
 else
 {
 if(b[j]%2!=0)
 {
 k++;
 cout<<b[j]<<' ';
 }
 j--;
 }
 }
 while(i>=0)
 {
 if(a[i]%2!=0)
 {

```



```

 k++;
 cout<<a[i]<<' ';
 }
 i--;
}

while(j>=0)
{
 if(b[j]%2!=0)
 {
 k++;
 cout<<b[j]<<' ';
 }
 j--;
}

if(k==0)cout<<"nu exista";
return 0;
}

```

### CAPITOLUL 5. Tablouri bidimensionale (matrice)

```

1. #include <iostream>
using namespace std;
int n, a[21][21], i, j, x;
int main()
{
 cin>>n>>x;
 for(i=1; i<=n; i++)
 for(j=1; j<=n; j++)
 if(i==1||j==1)a[i][j]=x;
 else a[i][j]=a[i-1][j]+a[i][j-1];
 for(i=1; i<=n; i++)
 {
 for(j=1; j<=n; j++)
 cout<<a[i][j]<<' ,';
 cout<<endl;
 }
 return 0;
}

```

```

2. #include <iostream>
using namespace std;
int m, n, a[11][11], i, j, max1, s;
int main()
{
 cin>>m>>n;
 for(i=1; i<=m; i++)
 for(j=1; j<=n; j++)
 cin>>a[i][j];

```

```

 for(i=1; i<=m; i++)
 {
 max1=a[i][1];
 for(j=2; j<=n; j++)
 if(a[i][j]>max1)
 max1=a[i][j];
 s=s+max1;
 }
 cout<<s;
 return 0;
 }
}

```

```

3. #include <iostream>
using namespace std;
int m, n, a[11][11], i, j, minc, s;
int main()
{
 cin>>m>>n;
 for(i=1; i<=m; i++)
 for(j=1; j<=n; j++)
 cin>>a[i][j];
 for(j=1; j<=n; j++)
 {
 minc=a[1][j];
 for(i=2; i<=m; i++)
 if(a[i][j]<minc)
 minc=a[i][j];
 s=s+minc;
 }
 cout<<s;
 return 0;
}

```

```

4. #include <iostream>
using namespace std;
int n, a[10][10], i, j;
int main()
{
 cin>>n;
 for(j=1; j<=n; j++)
 cin>>a[1][j];
 for(i=2; i<=n; i++)
 {
 for(j=1; j<=n; j++)
 a[i][j]=a[i-1][j+1];
 a[i][n]=a[i-1][1];
 }
 for(i=1; i<=n; i++)
 {
 for(j=1; j<=n; j++)
 cout<<a[i][j]<<' ,';
 cout<<endl;
 }
 return 0;
}

```

```

#include <iostream>
using namespace std;
int n,a[10][10],x,i,j,nr;
int main()

cin>>n;
for(i=1; i<=n; i++)
 for(j=1; j<=n; j++)
 cin>>a[i][j];

cin>>x;
for(i=2; i<n; i++)
{
 if(a[i][1]%x==0)
 nr++;
 if(a[i][n]%x==0)
 nr++;
 if(a[1][i]%x==0)
 nr++;
 if(a[n][i]%x==0)
 nr++;
}
if(a[1][1]%x==0)nr++;
if(a[1][n]%x==0)nr++;
if(a[n][1]%x==0)nr++;
if(a[n][n]%x==0)nr++;
cout<<nr;
return 0;
}

```

```

6. #include <iostream>
using namespace std;
int m,n,a[51][51],i,j,ok,afis;
int main()
{
 cin>>m>>n;
 for(i=1; i<=m; i++)
 for(j=1; j<=n; j++)
 cin>>a[i][j];
 for(i=1; i<=m; i++)
 {
 ok=1;
 for(j=2; j<=n&&ok; j++)
 if(a[i][j]!=a[i][1])
 ok=0;
 if(ok)
 {
 cout<<i<<' ',<<a[i][1]<<endl;
 afis=1;
 }
 }
 if(!afis)cout<<"nu exista";
 return 0;
}

```

```

7. #include <iostream>
using namespace std;
int m,n,a[51][51],i,j,ok,nr;
int main()
{
 cin>>m>>n;
 for(i=1; i<=m; i++)
 for(j=1; j<=n; j++)
 cin>>a[i][j];
 for(j=2; j<=n; j++)
 {
 ok=1;
 for(i=1; i<=m&&ok; i++)
 if(a[i][j]!=a[i][1])
 ok=0;
 if(ok)nr++;
 }
 cout<<nr;
 return 0;
}

```

```

8. #include <iostream>
using namespace std;
int n,a[16][16],i,j,x=1;
int main()
{
 cin>>n;
 for(i=1; i<=n; i++)
 for(j=1; j<=n; j++)
 if(i%2!=0)a[i][j]=x++;
 else a[i][n+1-j]=x++;
 for(i=1; i<=n; i++)
 {
 for(j=1; j<=n; j++)
 cout<<a[i][j]<<' ';;
 cout<<endl;
 }
 return 0;
}

```

```

9. #include <iostream>
using namespace std;
int n,a[16][16],i,j,x=1;
int main()
{
 cin>>n;
 for(i=1; i<=(n+1)/2; i++)
 {
 for(j=i; j<=n+1-i; j++)
 a[i][j]=x++;
 for(j=i+1; j<=n+1-i; j++)
 a[j][n+1-i]=x++;
 for(j=n-i; j>=i; j--)

```

```

 a[n+1-i][j]=x++;
 for(j=n-i; j>i; j--)
 a[j][i]=x++;
 }
 for(i=1; i<=n; i++)
 {
 for(j=1; j<=n; j++)
 cout<<a[i][j]<<' ';;
 cout<<endl;
 }
 return 0;
}

```

## CAPITOLUL 6. Șiruri de caractere

### 1. Varianta 1

```

#include <fstream>
#include<cstring>
using namespace std;
ifstream fin („sir.in");
ofstream fout („sir.out");

int main()
{char s[201],c[51],sn[201]="",*p;
int nr_cuv=0,i,j;
fin.getline(s,201);fin.
getline(c,51);
for(i=0;i<strlen(s);i++)
 if(s[i]==c[0]&& (i==0||s[i-1]!=' '))
 {
 nr_cuv++;
 j=i+1;
 while(s[j]!=' ',&&j<strlen(s))
 j++;
 strcpy(s+i,s+j); //eliminarea
 cuvântului cu prefix comun din text
 }
 fout<<s<<"\n"<<nr_cuv<<"\n";
 return 0;
}

```

### Varianta 2

```

#include <fstream>
#include<cstring>
using namespace std;
ifstream fin („sir.in");
ofstream fout („sir.out");

int main()
{char s[201],c[51],sn[201]="",*p;
int nr_cuv=0;

```

```

 fin.getline(s,201);fin.
 getline(c,51);
 p=strtok(s," ");
 while(p)
 {
 if(p[0]==c[0])//prima litera
 reprezinta un prefix al sirului
 nr_cuv++;
 else
 strcat(sn,p);
 strcat(sn," ");
 p=strtok(NULL," ");
 }
 sn[strlen(sn)-1]=NULL;
 strcpy(s,sn);
 fout<<s<<"\n"<<nr_cuv<<"\n";
 return 0;
}

```

### 2.

```

#include <iostream>
#include<cstring>
using namespace std;
int main()
{ char t[51],tnou[160]="",unu[]="unu",doi[]="doi",aux[51];int i;
cin>>t;
for(i=0;i<strlen(t);i++)
{
 if(t[i]=='1')
 {strcpy(aux,t+i+1);
 strcpy(t+i+1,unu);
 strcat(t,aux);
 }
 if(t[i]=='2')
 {strcpy(aux,t+i+1);
 strcpy(t+i+1,doi);
 strcat(t,aux);
 }
}
cout<<t;
return 0;
}

```

### 3.

```

#include <iostream>
#include<cstring>
#include<cctype>
using namespace std;
int n=0;

```

```

void numar(char s[])// adauga cifrele
din sirul s la numarul n

```

```

{int i;
 for(i=0;i<strlen(s);i++)
 if(isdigit(s[i])!=0)
 n=n*10+(s[i]-'0');
}
int main()
{char s1[251],s2[251];
 cin>>s1>>s2;
 numar(s1);numar(s2);
 cout << n;
 return 0;
}

```

4.

```

#include <fstream>
#include<cstring>
using namespace std;
ifstream fin(„cuvinte.in”);
ofstream fout(„cuvinte.out”);

int main()
{char prim[31],linie[31],ultim[31];
 int n,i;

 fin>>n; fin>>linie;
 strcpy(prim,linie);strcpy(ultim,linie);
 for(i=1;i<n;i++)
 {
 fin>>linie;
 if(strcmp(linie,prim)<0)
 strcpy(prim,linie);
 if(strcmp(linie,ultim)>0)
 strcpy(ultim,linie);
 }
 fout<<prim<<“\n”<<ultim;
 return 0;
}

```

5.

```

#include <iostream>
#include<cstring>
using namespace std;
int main()
{ char t[201],tnou[201],*cuv,
 separatori[]=",.?!",voc[]="aeiou";
 cin.getline(t,201);strcpy(tnou,"");
 cuv=strtok(t,separatori);
 while(cuv)
 {
 if(strchr(voc,cuv[0])!=NULL&&strchr
 (voc,cuv[strlen(cuv)-1])!=NULL)

```

```

 {strcat(tnou,cuv);strcat(tnou," ");}
 cuv=strtok(NULL,separatori);
 }
 if(strlen(tnou)>0)
 {tnou[strlen(tnou)-
1]=NULL;cout << tnou;}
 else
 cout<<“Nu exista”;
 return 0;
}

```

6.

```

#include <iostream>
#include<cstring>
#include<cctype>
using namespace std;

char e[201],operatori[]="-*/",
paranteze[]="[(,;";
int main()
{int i,j,corect=1;
 cin>>e;
 if(strlen(e)<3)
 corect=0;
 if(!islower(e[0])&&!isdi-
git(e[0])&&!strchr(operatori,e[0])&&
!strchr(paranteze,e[0]))
 corect=0;
 for(i=1;i<strlen(e)&&corect==1;i++)
 {if(!islower(e[i])&&!isdi-
git(e[i])&&!strchr(operatori,e[i])&&
!strchr(paranteze,e[i]))
 corect=0;

 if(islower(e[i])&&islower(e[i-1]))
 corect=0;

 if(isdigit(e[i])&&isdigit(e[i-1]))
 corect=0;
 if(islower(e[i])&&isdi-
git(e[i-1])||islower(e[i-1])&&
isdigit(e[i]))
 corect=0;
 if(strchr(operatori,e[i])&&
strchr(paranteze,e[i-1]))
 corect=0;
 if(strchr(operatori,e[i])&&
strchr(operatori,e[i-1]))
 corect=0;
 }
 cout <<corect;
 return 0;
}

```

## SPECIALIZAREA ȘTIINȚELE NATURII

## TEZA 1

## Subiectul I

1. d    2. b    3. b    4. b    5. d

## Subiectul II

4. a) Algoritmul calculează cifra de control a unui număr natural. Se va afișa 3.

b) Pentru orice număr nenul divizibil cu 9 se va afișa 9. Cel mai mic număr este 18, cel mai mare număr este 99.

c) #include &lt;iostream&gt;

using namespace std;

int x, y;

int main()

```
{
 cin>>x;
 while(x>9)
 {
 y=0;
 while(x>0)
 {y=y+x%10;
 x=x/10;
 }
 x=y;
 }
 cout<<y;
 return 0;
}
```

d) Se știe că cifra de control a unui număr se poate determina în funcție de restul împărțirii numărului la 9.

citește x (x număr natural)

dacă  $x \% 9 = 0$  atunci $x \leftarrow 9$ 

altfel

 $x \leftarrow x \% 9$ 

scrie x

```
5. per=sqrt((x[n]-x[1])*(x[n]-x[1])+(y[n]-y[1])*(y[n]-y[1]));
 for(i=2;i<=n;i++)
 per+=sqrt((x[i]-x[i-1])*(x[i]-x[i-1])+(y[i]-y[i-1])*(y[i]-y[i-1]));
```

```
6. Două tablouri pot fi: int x[]={20, 10, 9, 7, 5, 1}, y[]={19, 10, 8, 5, 1};
```

## Subiectul III

1. Vom descompune n în factori primi, algoritmul va determina suma factorilor primi.

citeste n

 $s \leftarrow 0; d \leftarrow 2$ cât timp ( $n \neq 1$ ) execută

```
{
 dacă ($n \% d == 0$) atunci
```

```
{
 cat timp ($n \% d == 0$) execută
 $n \leftarrow n / d$;
 $s \leftarrow s + d$;
 }
 $d \leftarrow d + 1$;
}
```

scrie s;

2. #include &lt;iostream&gt;

using namespace std;

int n, v[101], i, j, k;

int main()

```
{
 cin>>n;
 for(i=1; i<=n; i++)
 cin>>v[i];
 for(k=1; k<=n; k++)
 {
 i=1; j=n-k+1;
 while(i<=k)
 {
 cout<<v[i]<<' ';;
 i++;
 }
 while(j<=n)
 {
 cout<<v[j]<<' ';;
 j++;
 }
 cout<<endl;
 }
 return 0;
}
```

3. a) Vom citi succesiv numerele din fișier, vom determina valoarea cerută chiar la citire (eficiența timp). Determinăm după câți pași va fi eliminat numărul curent. Numărul maxim de pași necesari este numărul de prelucrări. Vom utiliza doar variabile simple (eficiența spațiu de memorie).

b) #include &lt;iostream&gt;

#include &lt;fstream&gt;

using namespace std;

ifstream fin(„bac.txt”);

int x, nrp, nrpmax, prim;

int main()

```
{
 while(fin>>x)
 {
 nrp=0;
 do{
 prim=1;
 if(x<=1 || x%2==0 && x!=2)
 prim=0;
 for(int i=3; i*i<=x; i=i+2)
```

```

 if (x%i==0) prim=0;
 if (prim==0) {
 nrp++;
 x++;
 }
} while (prim==0);
 if (nrp+1>nrpmax) nrpmax=nrp+1;
}
cout<<nrpmax;
return 0;
}

```

## TEZA 2

### Subiectul I

1. d    2. d    3. a    4. c    5. a

### Subiectul II

1. a) Algoritmul numără câte dintre valorile citite sunt puteri ale lui 2. Pentru valorile date se va afișa 3, deoarece sunt 3 puteri ale lui 2 (4, 8, 16).

b) Oricare patru numere care nu sunt puteri ale lui 2. Exemplu: 12 15, 13, 45

```

c) #include <iostream>
using namespace std;
int n,x,y;
int main()
{
 cin>>n;
 while (n>0)
 {
 cin>>x;
 while (x%2==0)
 x=x/2;
 if (x==1)
 y++;
 n--;
 }
 cout<<y;
 return 0;
}

```

d) Vom utiliza structura repetitivă „pentru – execută” citește n (n număr natural) y←0

```

pentru i←1, n execută
 citește x
 cât timp x%2=0 execută
 x←x/2
 dacă x=1 atunci
 y←y+1

```

scrie y

2. XM==0&&XN==0 || YM==0&&YN==0

3. C=(80, 78, 15, 15, 12, 9, 8, 7, 3, 3)

### Subiectul III

1. Numerele naturale care au exact trei divizori sunt pătratele perfecte de numere prime. Vom număra câte pătratele perfecte de numere prime sunt în intervalul [a,b].

```

citeste a,b
nr=0;
x←√a
y←√b
daca x*x<a atunci x←a+1
pentru i←x,y execută

```

```

{
 prim←1;
 daca (i<=1) atunci prim←0;
 pentru j←2,i/2 execută
 daca (i%j==0) prim←0;
 daca (prim==1) atunci

```

```

 nr←nr+1;
 }

```

scrie nr;

2. Vom parcurge vectorul și vom număra valorile egale cu valoarea curentă, fiecare valoare numărată va fi modificată.

```

#include <iostream>
using namespace std;
int n, v[51], i, j, k;
int main()
{
 cin>>n;
 for (i=1; i<=n; i++)
 cin>>v[i];
 for (i=1; i<=n; i++)
 if (v[i]!=-1)
 {
 cout<<v[i]<<' ';
 k=1;
 for (j=i+1; j<=n; j++)
 if (v[i]==v[j])
 {
 k++;
 v[j]=-1;
 }
 cout<<k<<endl;
 v[i]=-1;
 }
 return 0;
}

```

3. a) Deoarece valorile din cele două șiruri sunt numere naturale cu cel mult două cifre, vom utiliza doi vectori de frecvență, un vector pentru șirul a și un vector pentru șirul b. Apoi vom determina  $\min\{fa[i]/fb[i], \text{cu proprietatea că } fb[i] \neq 0, i=0,99\}$ . Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar.

```

b) #include <iostream>
#include<fstream>
using namespace std;
ifstream fin("bac.txt");
int fa[100],fb[100],i,x,nrs,n,m;
int main()
{
 fin>>n>>m;
 for(i=1;i<=n;i++)
 {fin>>x;
 fa[x]++;
 }
 for(i=1;i<=m;i++)
 {fin>>x;
 fb[x]++;
 }
 nrs=0;
 for(i=0;i<=99;i++)
 if(fb[i]!=0)
 if(nrs==0||fa[i]/fb[i]<nrs)
 nrs=fa[i]/fb[i];
 cout<<nrs;
 return 0;
}

```

**TEZA 3****Subiectul I**

1. c    2. c    3. b    4. a    5. a

**Subiectul II**

1. Algoritmul construiește cel mai mare număr care se poate forma utilizând toate cifrele lui x.

a) Se va afișa 7721.

b) Două numere care conțin cifrele 2, 3, 4 de același număr de ori pot fi 3324, 2343.

c) #include<iostream>

using namespace std;

int x,y,cx,i;

int main()

{cin>>x;

y=0;

for(i=9;i>=0;i--)

{

cx=x;

while(cx)

{if(cx%10==i)

y=y\*10+i;

cx=cx/10;

}

}

cout<<y;

return 0;

}

d) Vom utiliza structura repetitivă cât „timp – execută” citește x

(x număr întreg)

```

y<-0; i<-9
cât timp i≥0 execută
 cx<-x
 cât timp cx>0 execută
 dacă cx%10=i atunci
 y<-y*10+i
 cx<-cx/10
 i<-i-1

```

scrie y

2. cin>>pre>>pim;

cout<<sqrt(pre\*pre+pim\*pim);

3. if(i==j) cout<<2<<' ';

else cout<<1<<' ';

**Subiectul III**

1. citește n,m

nr<-0;

pentru i<-n,m executa

{

x<-i;

repetă

{

nr<-nr+1;

x<-x/10;

}pană când x=0;

}

scrie nr

2. Deoarece tabloul este ordonat, toate elementele egale sunt pe poziții consecutive.

#include <iostream>

using namespace std;

int n, v[101],w[101],i,j,k;

int main()

{

cin>>n;

for(i=1; i<=n; i++)

cin>>v[i];

w[1]=v[1];

k=1;

for(i=2; i<=n; i++)

if(v[i]!=v[i-1])

{

k++;

w[k]=v[i];

}

n=k;

for(i=1;i<=k;i++)

v[i]=w[i];

for(i=1;i<=k;i++)

cout<<v[i]<<' ';

return 0;

}

3. a) Vom utiliza un vector de frecvență. Deoarece numerele sunt întregi de două cifre, vom aduna

fiecare număr cu 99, astfel încât să lucrăm doar cu valori pozitive din intervalul [0,198]. La afișare avem grijă să scădem 99.

```
b) #include <iostream>
#include <fstream>
using namespace std;
ifstream fin("bac.txt");
int x, vf[200], i, vfmax;
int main()
{
 while(fin >> x)
 {
 vf[x+99]++;
 }
 for(i=-99; i<=99; i++)
 if(vf[i+99]>vfmax)
 vfmax=vf[i+99];
 for(i=-99; i<=99; i++)
 if(vf[i+99]==vfmax)
 cout<<i<<' ';
 return 0;
}
```

#### TEZA 4

##### Subiectul I

1. d    2. b    3. d    4. d    5. b

##### Subiectul II

1. a) algoritmul determină cel mai mare divizor al lui  $n$ , diferit de  $n$ , se va afișa 13;  
b) putem alege orice număr prim de o cifră 2, 3, 5, 7;

```
c) #include <iostream>
using namespace std;
int a, b, n, c;
int main()
{ cin >> n;
 a=1; b=n/2;
 while(a!=0 && b>0)
 { c=n;
 while(c>b)
 c=c-b;
 a=c;
 b=b-1;
 }
 cout<<b+1;
 return 0;
}
```

- d) A doua structură **cat timp** determină restul împărțirii lui  $n$  la  $b$

citește  $n$  ( $n$  număr natural nenul)

```
a ← 1
b ← [n / 2]
cât timp a ≠ 0 și b > 0 execută
{ c ← n % b
 a ← c
 b ← b - 1
}
b ← b + 1
scrie b
```

2.

```
int i, nr=0; float media;
for(i=0; i<200; i++)
{ if(mat[i]>=5 && lrom[i]>=5 && inf[i]>=5)
 { media=(mat[i]+lrom[i]+
 inf[i])/3.0;
 if(media>=7)
 nr++;
 }
}
cout<<nr;
3.
if (j==0 || j==6) cout<<'*'<<" ";
else
 if ((i<=3) && (j+i==6 || i==j))
 cout<<'*'<<" ";
 else cout<<'-'<<" ";
cout<<endl;
```

##### Subiectul III

1. citește  $n$  ( $n$  număr natural nenul)

găsit ← 0

```
dacă n % 2 ≠ 0
 atunci
 scrie 0
 altfel
 a ← n % 2
 n = [n / 2]
 cât timp n ≠ 0 și găsit = 0 execută
 { b ← n % 2
 n ← [n / 2]
 dacă a = b
 atunci
 scrie 0
 găsit ← 1
 }
 a ← b
```

```
■
dacă găsit = 0
 atunci
 scrie 1
 ■
```

```
2. #include <iostream>
using namespace std;
int main()
{ int n, k, p[1000], i, a, b, r;
 for(i=1; i<n; i++)
 {
 a=i; b=n;
 while(b)
 { r=a%b;
 a=b; b=r;
 }
 if(a==1)
 {k++;

```

```

 p[k]=i;}
}
cout<<k<<endl;
for(i=1;i<=k;i++)
 cout<<p[i]<<" ";
return 0;}

```

3. a) Citim k și apoi succesiv valorile din fișier, calculând pentru fiecare frecvență de apariție. Parcurgem descrescător numerele de la 99 la 0 adunând frecvențe, când suma este cel puțin k, ultimul număr pentru care am adunat frecvența este numărul căutat. Dacă în fișier sunt mai puțin de k valori se va afișa -1. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple și un vector de frecvență cu 100 de elemente.

3. b)

```

#include <iostream>
#include<fstream>
using namespace std;
ifstream fin("bac.txt");
int x,k,fr[100],i,s;
int main()
{
 fin>>k;
 while(fin>>x)
 fr[x]++;

 for(i=99; i>=0; i--)
 {
 s=s+fr[i];
 if(s>=k)
 {
 cout<<i;
 return 0;
 }
 }
 cout<<-1;
 return 0;
}

```

#### TEZA 5

##### Subiectul I

1. c    2. b    3. c    4. d    5. d

##### Subiectul II

- algoritmul determină cel mai mare divizor a n numere citite succesiv, se va afișa 2.
- putem alege oricare trei numere prime între ele 3, 5, 4.

```

c) #include<iostream>
using namespace std;
int n,i,r,d,x;
int main()

```

```

{ cin>>n;
d=0;
for(i=1;i<=n;i++)
{cin>>x;
if(d==0)d=x;
else
do{r=x%d;x=d;d=r;}while(r);
d=x;
}
cout<<d;
return 0;
}

```

d) citește n (n număr natural)

d←0

```

pentru i←1, n execută
 citește x
 dacă d=0 atunci
 d←x
 altfel
 r←x%d
 x←d
 d←r
 cât timp r>0 execută
 r←x%d
 x←d
 d←r
 d←x

```

scrie d

2. Se determină numărul de elevi înscriși la fiecare cerc și apoi se determină valoarea maximă dintre cele 3 numere.

```

int i,nr_i=0,nr_p=0,nr_r=0,
nr_max=0;
for(i=0;i<100;i++)
 if(cercuri[i]=='i')
 nr_i++;
 else
 if(cercuri[i]=='p')
 nr_p++;
 else
 nr_r++;
if(nr_i>nr_p)
 nr_max=nr_i;
else
 nr_max=nr_p;
if(nr_max<nr_r)
 nr_max=nr_r;
cout<<nr_max;

```

3.

```

{ if(i==6||i==1||j==6)
 cout<<1<<" ";
else

```

```

if(j<=2)
 cout<<j<<" ";
else
 if(j>=i&&j<=7-i)
 {if(i<=3)
 cout<<i<<" ";}
 else
 if(i>3&&j<=4)
 cout<<7-i<<" ";
 else
 cout<<7-j<<" ";
cout<<endl;
}

```

### Subiectul III

1. Parcurgem cifrele lui  $n$  de la stânga la dreapta. Când găsim o cifră mai mică decât cifra dată o inserăm.

```

citeste n, c
cn<-n;p<-1;ok<-0;
cat timp (n>9)executa
 n<-n/10;
 p<-p*10;
n<-0;
cat timp (p>0)executa
 cif<-(cn/p)%10;
 p<-p/10;
 daca (c>cif&&!ok)atunci
 n<-n*10+c;
 ok<-1;
 n<-n*10+cif;
daca(!ok)atunci
 n<-n*10+c;

```

```

2.
int main()
{ int n,v[1000],i,j;
 cin>>n;
 for(i=1; i<=n;i++)
 cin>>v[i];
 for(i=1; i<n;)
 if(v[i]==v[i+1])
 { for(j=i+2; j<=n; j++)
 v[j-1]=v[j];
 n--;
 }
 else i++;
 cout<<n;
 for(i=1; i<=n;i++)
 cout<<v[i]<<" ";
 return 0;
}

```

3. a) Vom citi numerele din fișier succesiv și vom determina intervalele asociate, pentru fiecare interval

asociat curent determinăm intersecția cu intervalul intersecție de până atunci. Dacă intervalul curent este  $[a, b]$ , atunci determinăm maximumul valorilor  $a$  și minimumul valorilor  $b$ . Dacă  $Maxa$  este mai mare decât  $Minb$ , afișăm mulțimea vidă. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple.

b)

```

#include <iostream>
#include <fstream>
using namespace std;
ifstream fin(„bac.txt”);
int x,y,a,b,maxa,minb,ok;
int main()
{
 fin>>x;
 a=x;
 ok=0;
 while(fin>>y)
 {
 if(y==x+1)b=y;
 else
 {
 if(ok==0)
 maxa=a,minb=b,ok=1;
 else
 {
 if(maxa<a)maxa=a;
 if(minb>b)minb=b;
 }
 a=y;
 }
 x=y;
 }
 if(ok==0)maxa=a,minb=b,ok=1;
 else
 {
 if(maxa<a)maxa=a;
 if(minb>b)minb=b;
 }
 if(minb>maxa)cout<<maxa<<
 <<minb;
 else cout<<„multimea vida”;
 return 0;
}

```

### TEZA 6

#### Subiectul I

1. a    2. c    3. c    4. b    5. c

#### Subiectul II

1. a) algoritmul afișează în ordine descrescătoare toate numerele din intervalul  $[a, b]$  cu proprietatea că

în reprezentarea în baza 2 numărul de cifre 1 este egal cu numărul de cifre egale cu 0. Se va afișa 12 10 9 \*.

b) cel mai mare număr este 34.

```
c) #include<iostream>
using namespace std;
int a,b,x,y,k;
int main()
{
 cin>>a>>b;
 for(x=b;x>=a;x--)
 {
 y=x;k=0;
 while(y>0)
 {
 k=k+1-2*(y%2);
 y=y/2;
 }
 if(k==0)
 cout<<x<<' ';;
 }
 cout<<'**';
 return 0;
}
```

d) citește a, b (a, b numere naturale,  $a \leq b$ )  
 $y \leftarrow 0$

```
pentru x ← b, a, -1 execută
 y ← x
 k ← 0
 dacă y > 0 atunci
 repetă
 k ← k+1-2*(y%2)
 y ← [y/2]
 până când y=0
 dacă k=0 atunci
 scrie x, ' '
scrie '**'
```

2.

```
int lungime[100],latime[100],
inaltime[100],nr_cuburi=0,i;
for(i=0;i<100;i++)
 if(lungime[i]==latime[i]&&
 latime[i]==inaltime[i])
 nr_cuburi++;
cout<<nr_cuburi;
```

```
3. {if(i%3+j%3==4) cout<<6<<" ";
 else cout<<i%3+j%3<<" ";
 cout<<endl;
}
```

### Subiectul III

1. citește a,b

$p \leftarrow 1; ca \leftarrow a; n \leftarrow 0;$

cât timp ( $ca > 9$ ) execută

$ca \leftarrow ca/10;$

$n \leftarrow n+1$

$p \leftarrow p*10;$

pentru  $i \leftarrow 0, n-1$  execută

dacă  $a=b$  și  $ok=0$  atunci

Scrie i

$ok \leftarrow 1$

altfel

$a \leftarrow (a\%10)*p+a/10$

dacă  $ok=0$  atunci

scrie -1

2. Cel mai mic divizor diferit de 1 a lui n este prim, vom căuta valoarea divizorului în vectorul v prin căutare binară. Dacă îl găsim se afișează poziția divizorului în vector, în caz contrar îl vom insera în v pe poziția furnizată de căutarea binară și afișăm vectorul.

int main()

```
{ int k,n,v[30],d,ls,ld,mij,ok=0, poz;
 cin>>k;
 for(i=1;i<=k;i++)
 cin>>v[i];
 d=n;
 for(int i=2;i*i<=n;i++)
 if(n%i==0){d=i;break;}
 ls=1;ld=k;
 while(ls<=ld&&ok)
 {
 mij=(ls+ld)/2;
 if(v[mij]==d){ok=1;poz=mij;}
 else if(v[mij]>d)ld=mij-1;
 else ls=mij+1;
 }
 if(!ok)
 {
 for(int i=k;i>=ls+1;i--)
 v[i+1]=v[i];
 v[ls+1]=d;k++;
 for(i=1;i<=k;i++)
 cout<<v[i]<<" ";
 }
 else
 cout<<poz;
}
```

3. a) Vom citi numerele din fișier succesiv și vom memora într-un vector de frecvență numărul de apariții a fiecărei cifre. Cifrele care apar în palindrom trebuie să apară de un număr par de ori, mai

puțin cifra din mijloc. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple și un vector cu 10 elemente.

```
b) #include <iostream>
#include<fstream>
using namespace std;
ifstream fin("bac.txt");
int x,fr[10],exista=1,i,j,mijloc,p,n;
int main()
{
 fin>>n;
 for(i=1; i<=n; i++)
 {
 fin>>x;
 fr[x]++;
 }
 //determin cifra din mijloc daca
 exista, trebuie sa fie una singura
 mijloc=-1;
 for(i=9; i>=1; i--)
 if(fr[i]%2==1)
 if(mijloc==-1)
 {
 mijloc=i;
 }
 else
 {
 exista=0;
 break;
 }
 if(exista)
 {
 for(i=9; i>=1; i--)
 {
 for(j=1; j<=fr[i]/2;
j++)
 cout<<i;
 }
 if(mijloc!=-1)cout<<mijloc;
 for(i=1; i<=9; i++)
 {
 for(j=1; j<=fr[i]/2;
j++)
 cout<<i;
 }
 }
 if(!exista)cout<<-1;
 return 0;
}
```

## TEZA 7

### Subiectul I

1. b    2. c    3. d    4. c    5. b

### Subiectul II

1a. 2664

1. b) 2 0 500 500 sau 4 3 334 103 533 133 (orice set de n numere, pentru care suma numerelor care conțin cifra y de un număr par de ori este egală cu 1000).

1. c)

```
#include <iostream>
using namespace std;
int main()
{
 int e,n,y,i,x,cx,nr;
 cin>>n>>y;
 e=0;
 for(i=1;i<=n;i++)
 {
 cin>>x;
 cx=x;
 nr=0;
 while(x!=0)
 {
 if(x%10==y)
 nr++;
 x=x/10;
 }

 if(nr%2==0 && nr>0)
 e=e+cx;
 }
 cout<<e;
 return 0;
}
```

1. d)

citește n, y (y număr natural,  $y < 10$ , n număr natural nenul)

$e \leftarrow 0$

$i \leftarrow 1$

cât timp  $i \leq n$  execută

citește x (număr natural nenul)

$cx \leftarrow x$

$nr \leftarrow 0$

cât timp  $x \neq 0$  execută

dacă  $x \% 10 = y$

atunci

$nr \leftarrow nr + 1$

$x \leftarrow [x/10]$

dacă  $nr \% 2 = 0$  și  $nr > 0$

atunci

$e \leftarrow e + cx$

$i \leftarrow i + 1$

scrie e

2.

int i,p,pmax,pmin,n;

pmin=pmax=

latura1[0]+latura2[0]+latura3[0];

for(i=1;i<n;i++)

```
{p=latura1[i]+latura2[i]+latura3[i];
if(p<pmin)
 pmin=p;
if(p>pmax)
 pmax=p;
}
```

```
cout<<pmin<<" "<<pmax;
```

3.

```
if(i==0)
 cout<<(char)('A'+j)<<" ";
else
 if(j==0)
 cout<<(char)('A'+i)<<" ";
 else
 if(i==j)
 cout<<"A ";
 else
 if(i+j==5)
 cout<<"F ";
else
 cout<<(char)('A'+i)<<" ";
```

### Subiectul III

1.

Citeste n

i ← 1

k ← 3

cât timp i ≤ n execută

j ← 1;

cât timp j ≤ k și i ≤ n

dacă i = n

atunci

termen = j

j ← j + 1

i ← i + 1

j ← k - 1

cât timp j > 1 și i ≤ n

dacă i = n

termen = j

j ← j - 1

i ← i + 1

k ← k + 3

scrie termen;

2.

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{int prodmax=0,i,j, n, x[100];
```

```
cin>>n;
```

```
for(i=0;i<n;i++)
```

```
cin>>x[i];
```

```
for(i=0;i<n-1;i++)
```

```
for(j=i+1;j<n;j++)
```

```
if(x[i]*x[j]>prodmax)
```

```
prodmax=x[i]*x[j];
```

```
cout<< prodmax;
```

```
return 0;
```

```
}
```

3. a) Programul citește numerele din fișier pe rând, verifică pentru fiecare număr dacă prima și ultima cifră sunt egale și în caz afirmativ reține frecvența numărului interior al numărului testat în vectorul f. Se afișează numerele interioare utilizând vectorul de frecvență. Programul este eficient ca timp de execuție deoarece obține numerele interioare la o singură parcurgere a fișierului, simultan cu citirea.

3. b)

```
#include <iostream>
```

```
#include<fstream>
```

```
using namespace std;
```

```
ifstream fin(„bac.in”);
```

```
int f[1000];
```

```
int main()
```

```
{ int x,uc,p,nr;
```

```
while(fin>>x)
```

```
{ p=1;nr=0;
```

```
uc=x%10;x=x/10;
```

```
while(x>9)
```

```
{ nr=nr+(x%10)*p;// nr-
```

```
număr interior al lui x
```

```
p*=10;
```

```
x/=10;
```

```
}
```

```
if(uc==x)
```

```
f[nr]++;
```

```
}
```

```
p=0;
```

```
for(x=999;x>=0;x--)
```

```
if(f[x]!=0)
```

```
{cout<<x<<" ";
```

```
p=1;
```

```
}
```

```
if(p==0)
```

```
cout<<"Nu exista";
```

```
return 0;
```

```
}
```

### TEZA 8

#### Subiectul I

1. c

2. c

3. d

4. d

5. c

#### Subiectul II

1. a) k=49 și q=35

1. b) a=2 și b=3

1. c)

```
#include <iostream>
using namespace std;
int main()
{
 int a,b,k,q,x,y;
 k=0;q=0;
 cin>>a>>b;
 for(x=1;x<=a;x++)
 for(y=1;y<=b;y++)
 if(y%x==0)
 k+=y;
 else
 q+=y;
 cout<<k<<" "<<q;
 return 0;
}
```

1. d)

citește a,b ( $a \leq b$ , a, b numere naturale) $k \leftarrow 0; q \leftarrow 0; x \leftarrow 1$ cât timp  $x \leq a$  executăpentru  $y \leftarrow 1, b$  executădacă  $y \% x = 0$ 

atunci

 $k \leftarrow k + y$ 

altfel

 $q \leftarrow q + y$  $x \leftarrow x + 1$ 

scrie k, " ", q

2.

int i;

for (i=1; i&lt;=200; i++)

if (loc[i]/supr[i]&gt;15)

ccut&lt;&lt;i&lt;&lt;" ";

2. cout&lt;&lt;i+j+6&lt;&lt;" ";

**Subiectul III**

1.

Citește x

 $nr \leftarrow 0$ pentru i  $\leftarrow 1, n-1$  execută

citește y

 $c \leftarrow y$  $ogl \leftarrow 0$ cât timp  $c \neq 0$  execută $ogl \leftarrow ogl * 10 + c \% 10$  $c \leftarrow [c / 10]$ dacă  $x == ogl$ 

atunci

 $nr \leftarrow nr + 1$  $x \leftarrow y$ 

scrie nr;

2.

#include&lt;iostream&gt;

using namespace std;

int main()

{int f[100],pt[100],i,c,nf=0,d,pf,

poz,ok=0,n,fp,p,v[51];

cin&gt;&gt;n;

for(i=1;i&lt;=n;i++)

cin&gt;&gt;v[i];

//nf- nr factori, f-factori primi,

p-puteri factori primi

fp=1;

c=v[1];

d=2;

//determin factorii primi si puterile

lor pt v[1] si ii

// memorez in vectorul f, puterile

lor le memorez in vectorul //pt

while(c&gt;1)

{pf=0;

while(c%d==0)

{

c/=d;

pf++;

}

if(pf!=0)

{

nf++;

f[nf]=d;

pt[nf]=pf;

}

d++;

}

poz=1;

int j;

for(i=2;i&lt;=n&amp;&amp;fp&gt;0;i++)

{c=v[i];ok=0;

// f[j] este factor prim comun

al lui v[1] si v[i]

for(j=poz;j&lt;=nf;j++)

if(c%f[j]==0)

{ pf=0;ok=1;

while(c%f[j]==0)

{pf++;

c/=f[j];

}

if(pf&lt;pt[j])

pt[j]=pf;

fp=f[j];p=pt[j];poz=j;

break;

}

if(ok==0)

{

fp=-1;p=0;

}

}

cout&lt;&lt;fp&lt;&lt;" "&lt;&lt;p;

return 0;

}

3. a) Programul este eficient ca timp de execuție deoarece generează direct numerele care trebuie afișate, alegând în ordine crescătoare primele 3 cifre impare  $c_1, c_2, c_3$ , palindromul afișat va fi  $c_1 c_2 c_3 c_3 c_2 c_1$ . Programul este eficient din punct de vedere al memoriei utilizate deoarece are un număr mic de variabile simple.

3. b)

```
#include<fstream>
using namespace std;
ofstream fout („bac.txt");
int main()
{
 int c1,c2,c3,nr;
 for(c1=1;c1<=9;c1=c1+2)
 for(c2=1;c2<=9;c2=c2+2)
 for(c3=1;c3<=9;c3=c3+2)
 { nr=c1*100000+c2*10000+c3*
 1000+c3*100+c2*10+c1;
 fout<<nr<<endl;
 }
 return 0;
}
```

## TEZA 9

### Subiectul I

1. c    2. c    3. b    4. a    5. b

### Subiectul II

1. a) 12, 15, 18, 21, 24, 27, 30, 33, 36;  
1 b. Orice set de numere a, b cu  $b - a = 2$ ,  $k = 2$ , pentru care a și b au prima cifră un număr par. De exemplu:  $a = 20$ ,  $b = 22$ ,  $k = 2$ , se afișează 20, 21, 22.

1. c)

```
#include <iostream>
using namespace std;
int main()
{int x,a,b,k,t,y;
cin>>a>>b>>k;
for(x=a;x<=b;x++)
{ y=0;t=x;
while(t!=0)
{ y=y*10+t%10;
t=t/10;
}
if(y%k==0)
cout<<x<<" ";
}
```

1. d)

citește a, b, k (numere naturale nenule,  $a \leq b$ ,  $k > 1$ )  
 $x \leftarrow a$

```
cât timp $x \leq b$ execută
y ← 0
t ← x
cât timp $t \neq 0$ execută
y ← $y * 10 + t \% 10$
t ← $t / 10$
dacă $y \% k = 0$
atunci
scrie x, " "
x ← x + 1
```

2.

```
int suma[11],suma_totala=0,
suma_prec[11],i;
for(i=1;i<=10;i++)
{ suma_totala= suma_totala+suma[i];
if(suma[i]>suma_prec[i])
cout<<i<<" ";
}
cout<<suma_totala;
3. if(i%2==0)
cout<<j<<" ";
else
cout<<2*i+j<<" ";
```

### Subiectul III

1.

Citește n

$m \leftarrow 1$

$t \leftarrow 1$

```
cât timp $t < n$ execută
s ← m+t
m ← t
t ← s
dacă $t = n$ atunci
t ← m+t
```

scrie m, t;

2.

```
int main()
{ int n,a[50],nr_pere-
chi=0,x,y,i,j,rest;
cin>>n;
for(i=0;i<n;i++)
cin>>a[i];
for(i=0;i<n-1;i++)
for(j=i+1;j<n;j++)
{ x=a[i];y=a[j];
do //se determina cmmdc al
numerelor x si y
{ rest=x%y;
x=y;y=rest;
}
```

```

while (rest!=0);
if (x==1)
 nr_perechi++;
}
cout<< nr_perechi;
return 0;
}

```

3. a) Programul citește numerele din fișier și determină pentru fiecare dintre benzi (formate din 7 numere citite consecutiv) cele mai mici două valori **min1** și **min2** ( $\text{min1} < \text{min2}$ ). La schimbarea unei benzi se afișează cele două minime din banda anterioară sau valoarea 0 (dacă nu există două minime diferite). Programul este eficient ca timp de execuție deoarece obține minimele și le afișează la o singură parcurgere a fișierului, simultan cu citirea. Programul este eficient din punct de vedere al memoriei utilizate deoarece memorează un număr minim de variabile.

3. b)

```

#include <iostream>
#include <fstream>
using namespace std;
ifstream fin("bac.txt");
int main()
{
 int n, i, min1, min2, x;
 fin>>n;
 min1=min2=1000;
 for (i=1; i<=n; i++)
 {
 fin>>x;
 if (x<min1)
 {
 min2=min1;
 min1=x;
 }
 else
 if (x<min2 && x>min1)
 min2=x;
 if (i%7==0)
 {
 if (min1!=1000 && min2!=1000 && min1!=min2)
 cout<<min1<<" "<<min2<<" ";
 else
 cout<<0<<" ";
 min1=min2=1000;
 }
 }
 return 0;
}

```

## TEZA 10

### Subiectul I.

1. c    2. c    3. d    4. b    5. c

### Subiectul II.

1. a) Algoritmul afișează produsul factorilor primi care apar la putere impară. Se va afișa 231.

b) cel mai mic număr este 25, cel mai mare este 81

c) #include<iostream>

using namespace std;

int n,p,m,x;

int main()

{

cin>>n;

p=1;

m=0;

x=2;

while (n>=x)

{

if (n%x==0)

{

m++;

n=n/x;

}

else

{

if (m%2!=0)

p=p\*x;

m=0;

x++;

}

}

if (m%2!=0) p=p\*x;

cout<<p;

return 0;

}

d) citește n (n număr natural, n>1)

$p \leftarrow 1$ ;  $m \leftarrow 0$ ;  $x \leftarrow 2$

dacă  $n \geq x$  atunci

repetă

dacă  $n \% x = 0$  atunci

$m \leftarrow m+1$ ;  $n \leftarrow [n/x]$

altfel

dacă  $m \% 2 \neq 0$  atunci

$p \leftarrow p \cdot x$

$m \leftarrow 0$ ;  $x \leftarrow x+1$

■

■

până când  $n < x$

■

dacă  $m \% 2 \neq 0$  atunci

$p \leftarrow p \cdot x$

■

scrie p

2.  $a == b \&\& a != c \ || \ a == c \&\& a != b \ || \ c == b \&\& a != c$

3.  $n=3$ ,  $m=3$ , cele două tablouri pot fi  $a=(2,4,6)$  și  $b=(3,5,7)$ .

## Subiectul III.

1. citește n (n număr natural)

```

pentru i ← 1, n execută
 citește x
 cx ← x; x ← x+1
 cât timp x%2=0 execută
 x ← [x/2]
 dacă x=1 atunci
 scrie cx,' '

```

2.

```

#include<iostream>
using namespace std;
int main()
{
 int n,v[100],i,j,k,ok;
 cin>>n;
 for(i=1; i<=n; i++)
 cin>>v[i];
 cin>>k;
 for(i=1; i<=k/2; i++)
 swap(v[i],v[k+1-i]);
 i=k+1;j=n;
 while(i<j)
 {
 swap(v[i],v[j]);
 i++;j--;
 }
 for(i=1; i<=n/2; i++)
 swap(v[i],v[n+1-i]);
 ok=1;
 for(i=1; i<=n/2; i++)
 if(v[i]!=v[n+1-i])
 ok=0;
 if(ok) cout<<"DA";
 else cout<<"NU";
 return 0;
}

```

3. a) Vom determina pentru fiecare număr printr-un algoritm eficient numărul de valori egale cu 0 de la sfârșitul lui  $x!$ , acest număr este egal cu puterea lui 5 în descompunerea în factori primi a lui  $x!$ . Determinăm maximul de zerouri și numărul de maxime printr-un algoritm clasic. Utilizăm doar variabile simple (complexitate spațiu de memorie), rezolvăm cerința în timpul citirii (complexitate timp).

b) #include &lt;iostream&gt;

#include&lt;fstream&gt;

using namespace std;

ifstream fin("bac.txt");

int x,zmax, nrzmax,nrz;

int main()

```

{
 while(fin>>x)
 {
 nrz=0;
 while(x)
 {
 nrz+=x/5;
 x=x/5;
 }
 if(nrz>nrzmax)
 {
 nrzmax=nrz;
 zmax=1;
 }
 else if(nrz==nrzmax) zmax++;
 }
 cout<<zmax<<' ,<<nrzmax;
 return 0;
}

```



## CUPRINS



### PARTEA I – BREVIAR

#### CAPITOLUL 1

|                                                                                    |   |
|------------------------------------------------------------------------------------|---|
| Reprezentarea algoritmilor în pseudocod. Elemente de bază ale limbajului C++ ..... | 5 |
|------------------------------------------------------------------------------------|---|

#### CAPITOLUL 2

|                            |    |
|----------------------------|----|
| Algoritmi elementari ..... | 15 |
|----------------------------|----|

#### CAPITOLUL 3

|                    |    |
|--------------------|----|
| Fișiere text ..... | 25 |
|--------------------|----|

#### CAPITOLUL 4

|                                          |    |
|------------------------------------------|----|
| Tablouri unidimensionale (vectori) ..... | 28 |
|------------------------------------------|----|

#### CAPITOLUL 5

|                                         |    |
|-----------------------------------------|----|
| Tablouri bidimensionale (matrice) ..... | 34 |
|-----------------------------------------|----|

#### CAPITOLUL 6

|                           |    |
|---------------------------|----|
| Șiruri de caractere ..... | 39 |
|---------------------------|----|

#### CAPITOLUL 7

|                                   |    |
|-----------------------------------|----|
| Structuri de date neomogene ..... | 44 |
|-----------------------------------|----|

#### CAPITOLUL 8

|                             |    |
|-----------------------------|----|
| Subprograme (Funcții) ..... | 50 |
|-----------------------------|----|

#### CAPITOLUL 9

|                         |    |
|-------------------------|----|
| Funcții recursive ..... | 55 |
|-------------------------|----|

#### CAPITOLUL 10

|                                                      |    |
|------------------------------------------------------|----|
| Metoda backtracking. Elemente de combinatorică ..... | 60 |
|------------------------------------------------------|----|

#### CAPITOLUL 11

|                                     |    |
|-------------------------------------|----|
| Elemente de teoria grafurilor ..... | 65 |
|-------------------------------------|----|

### PARTEA a II-a – TEZE

|                                            |     |
|--------------------------------------------|-----|
| Specializarea Matematică-Informatică ..... | 79  |
| Specializarea Științe ale Naturii .....    | 159 |

### PARTEA a III-a – TEZE

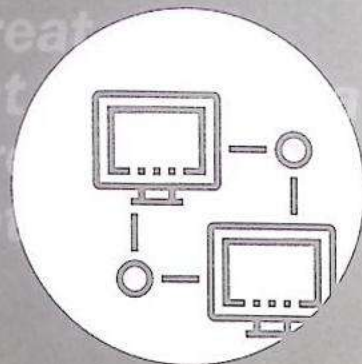
|               |     |
|---------------|-----|
| SOLUȚII ..... | 191 |
|---------------|-----|

## BIBLIOGRAFIE



- *Programarea în limbajul C/C++ pentru liceu*, vol. 1, Emanuela Cerchez, Marinel Șerban, Editura Polirom, Iași
- *Programarea în limbajul C/C++ pentru liceu*, vol. 2, Emanuela Cerchez, Marinel Șerban, Editura Polirom, Iași
- *Programarea în limbajul C/C++ pentru liceu*, vol. 3, Emanuela Cerchez, Marinel Șerban, Editura Polirom, Iași
- *Fundamentele Programării, Culegere de probleme pentru clasa a IX-a*, Dana Lica, Mircea Pașoi, Editura L&S Info-Mat, București
- *Fundamentele Programării, Culegere de probleme pentru clasa a X-a*, Dana Lica, Mircea Pașoi, Editura L&S Info-Mat, București
- *Fundamentele Programării, Culegere de probleme pentru clasa a XI-a*, Dana Lica, Mircea Pașoi, Editura L&S Info-Mat, București
- *Informatică Intensiv C++, Manual pentru Clasa a IX-a*, Mariana Miloșescu, Editura Didactică și Pedagogică, București
- *Informatică Intensiv C++, Manual pentru Clasa a X-a*, Mariana Miloșescu, Editura Didactică și Pedagogică, București
- *Informatică Intensiv C++, Manual pentru Clasa a XI-a*, Mariana Miloșescu, Editura Didactică și Pedagogică, București
- Subiecte Bacalaureat 2003, 2004, 2005, 2006
- Variante de Bacalaureat 2007, 2008, 2009
- [subiecte.edu.ro](http://subiecte.edu.ro)- subiecte Bacalaureat
- subiecte admitere Facultatea de Informatică, Iași
- subiecte admitere Facultatea de Matematică-Informatică, București
- subiecte admitere Facultatea de Matematică-Informatică, Cluj-Napoca
- [pbinfo.ro](http://pbinfo.ro)
- [campion.edu.ro/arhiva](http://campion.edu.ro/arhiva)





Lucrarea de față se dorește a fi un ghid de pregătire individuală a elevilor de liceu pentru proba de Informatică a examenului de Bacalaureat – varianta C++. În acest sens, cele unsprezece capitole de teorie prezintă un breviar al noțiunilor studiate la disciplina Informatică, prezente în programa de Bacalaureat, urmat de modele de teste propuse prin a căror rezolvare se poate realiza aprofundarea acestor noțiuni.

După parcurgerea noțiunilor de teorie, cartea se continuă cu teze întocmite după modelul examenului de Bacalaureat, atât pentru specializarea Matematică-Informatică, cât și pentru specializarea Științe ale Naturii.

Pentru a veni în sprijinul celor care vor folosi această carte, problemele și tezele au fost rezolvate integral, iar programele au fost verificate în Code::Blocks.

Ghidul poate fi utilizat în aceeași măsură și ca auxiliar didactic la orele de Informatică.

Sperăm că acest ghid se va dovedi un instrument util elevilor, dar și profesorilor care îl folosesc în pregătirea examenului de Bacalaureat.

Autoarele

pret  
cu TVA  
**35**  
lei

ISBN 978-973-47-3115-2



www.edituraparelela45.ro

BAC. INFORMATICA. LIMB  
AJUL C++  
9789734731152  
35,00 LEI/UM>



EDITURA PARALELA45  
EDUCATIONAL

