

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/320517919>

# Noțiuni de programare în limbajul C++

Book · October 2012

---

CITATION

1

READS

28,201

2 authors:



Adrian Runceanu

University of Craiova

26 PUBLICATIONS 39 CITATIONS

[SEE PROFILE](#)



Mihaela Runceanu

Colegiul National "Ecaterina Teodoroiu"

4 PUBLICATIONS 2 CITATIONS

[SEE PROFILE](#)

**ADRIAN RUNCEANU**

**MIHAELA RUNCEANU**

# **NOTIUNI DE PROGRAMARE ÎN LIMBAJUL**

**C++**

**EDITURA ACADEMICA BRÂNCUȘI DIN TÂRGU-JIU  
2012**

ADRIAN RUNCEANU

MIHAELA RUNCEANU

# **NOȚIUNI DE PROGRAMARE ÎN LIMBAJUL C++**

**Editura Academica Brâncuși**

**ISBN 978-973-144-550-2**

**Descrierea CIP a Bibliotecii Naționale a României**

**RUNCEANU, ADRIAN**

**Noțiuni de programare în limbajul C++** / Adrian Runceanu, Mihaela Runceanu – Târgu-Jiu: Academica Brâncuși, 2012

Bibliogr.

ISBN 978-973-144-550-2

**Coperta și design:**

Silviu Runceanu

**[www.silviu.runceanu.ro](http://www.silviu.runceanu.ro)**

ADRIAN RUNCEANU

MIHAELA RUNCEANU

# **NOȚIUNI DE PROGRAMARE ÎN LIMBAJUL C++**

**Editura Academica Brâncuși**

**Referenți științifici:**

**Conf. univ. dr. ing. Florin Grofu**, Universitatea Constantin Brâncuși din Târgu-Jiu, Facultatea de Inginerie, Departamentul de Automatică, Energie și Mediu

**Şef lucrări dr. ing. Marian Popescu**, Universitatea Constantin Brâncuși din Târgu-Jiu, Facultatea de Inginerie, Departamentul de Automatică, Energie și Mediu

*Dedicăm această carte:*

*Părintilor noștri, (Georgeta și Vasile Runceanu, Cornelia și Vasile Irod) care ne-au îndrumat pașii în viață.*

*Copiielor noștri, Alexandra și Mihai, cu foarte multă dragoste.*

# Cuvânt înainte

Cartea **Noțiuni de programare în limbajul C++** se adresează atât studenților cât și elevilor care doresc să studieze elementele de bază ale algoritmicii și programării calculatoarelor, dar și celor care doresc să-și perfecționeze tehnica dezvoltării și optimizării programelor.

Cartea este structurată în 9 capitole, în care primele 8 capitole fiind concepute într-o manieră didactică astfel încât să fie accesibile atât începătorilor cât și celor care sunt deja familiarizați cu elementele de programare. Ultimul capitol cuprinde teste de verificare a competențelor dobândite prin parcursarea întregii cărți. Fiecare capitol conține o parte teoretică, probleme rezolvate, teste grilă și probleme propuse spre rezolvare.

Am ales limbajul de programare C++ deoarece este implementat pe majoritatea platformelor de calcul existente azi și este cel mai popular limbaj de programare pentru scrierea de software de sistem. Totodată, sintaxa limbajului C++ a stat la baza multor limbiage create ulterior și foarte populare azi: Java, JavaScript, C#.

Cartea pune la dispoziția cititorului o parte introductivă ce conține un foarte scurt istoric al calculatoarelor, elemente de bază despre arhitectura și funcționarea unui sistem de calcul și noțiuni de algoritmică.

În primele 8 capitole ale cărții se studiază pe larg elementele de bază ale limbajului C++, tablourile unidimensionale și bidimensionale, sirurile de caractere, structurile și uniunile, subprogramele, pointerii, fișierele, sistemele de intrare/ieșire și funcțiile din bibliotecile standard. Toate exemplele prezentate în această carte au fost implementate cu ajutorul compilatorului C++ și testate pe date diferite.

Considerăm că acestă prezentare a noțiunilor programare în limbajul C++ este perfectibilă și în acest scop aşteptăm orice sugestii, comentarii și sfaturi din partea cititorilor și a tuturor celor care sunt interesați de conținutul acestei cărți, la adresele [adrian\\_r@utgjiu.ro](mailto:adrian_r@utgjiu.ro), [mihaela@utgjiu.ro](mailto:mihaela@utgjiu.ro).

Octombrie 2012

Autorii

# Cuprins

Introducere	9
Capitolul 1 Elemente de bază ale limbajului C++	31
1.1. Tipuri de date. Variabile. Constante	31
1.1.1. Cuvinte cheie	31
1.1.2. Tipuri de date	32
1.1.3. Constante	35
1.1.4. Variabile	37
1.2. Operatorii limbajului C++	40
1.3. Instrucțiunile limbajului C++	51
1.4. Probleme rezolvate	67
1.4.1. Enunțuri	67
1.4.2. Soluții propuse	69
1.5. Probleme propuse spre rezolvare	78
1.6. Teste grilă	95
1.7. Răspunsuri la întrebările grilă	117
Capitolul 2 Tablouri unidimensionale și bidimensionale	119
2.1. Tablouri în limbajul C++	119
2.1.1. Tablouri unidimensionale	119
2.1.2. Tablouri bidimensionale	122
2.1.3. Tablouri multidimensionale	125
2.2. Probleme rezolvate	127
2.2.1. Enunțuri	127
2.2.2. Soluții propuse	129
2.3. Probleme propuse spre rezolvare	139
2.4. Teste grilă	147

2.5.	Răspunsuri la întrebările grilă	160
<b>Capitolul 3</b>	<b>Şiruri de caractere. Structuri și uniuni</b>	<b>161</b>
3.1.	Şiruri de caractere	161
3.2.	Probleme rezolvate	161
3.2.1.	Enunțuri	167
3.2.2.	Soluții propuse	167
3.3.	Probleme propuse spre rezolvare	171
3.4.	Structuri și uniuni	178
3.4.1.	Definirea structurilor	178
3.4.2.	Declararea variabilelor de tip structură	180
3.4.3.	Accesarea elementelor unei structuri	180
3.4.4.	Tablouri de structuri	181
3.4.5.	Pointeri la structuri	181
3.4.6.	Tipuri utilizator	182
3.4.7.	Uniuni	182
3.5.	Probleme rezolvate	183
3.5.1.	Enunțuri	183
3.5.2.	Soluții propuse	184
3.6.	Probleme propuse spre rezolvare	191
3.7.	Teste grilă	199
3.8.	Răspunsuri la întrebările grilă	209
<b>Capitolul 4</b>	<b>Functii</b>	<b>211</b>
4.1.	Declarare. Apel. Prototip	211
4.1.1.	Declararea funcției	211
4.1.2.	Apelul unei funcții	212
4.1.3.	Prototipul funcției	212
4.1.4.	Parametri formali și actuali	214
4.1.5.	Variabile locale și variabile globale	215

4.1.6. Variabilele statice și dinamice	218
4.1.7. Funcții matematice	220
4.1.8. Apelul prin valoare	225
4.1.9. Apelul prin referință	228
4.1.10. Pointeri către funcții	232
4.1.11. Argumentele liniei de comandă	234
3.1. Probleme rezolvate	236
3.1.1. Enunțuri	236
3.1.2. Soluții propuse	237
3.2. Probleme propuse spre rezolvare	248
3.3. Teste grilă	260
3.4. Răspunsuri la întrebările grilă	276
<b>Capitolul 5 Pointeri</b>	<b>277</b>
5.1. Declarație	277
5.2. Operații specifice pointerilor	279
5.3. Arithmetica pointerilor	280
5.3.1. Pointeri și tablouri	281
5.3.2. Echivalențe de scriere	283
5.3.3. Folosirea pointerilor de tip void	285
5.3.4. Expresii compacte cu pointeri	285
5.3.5. Tablouri cu pointeri	287
5.4. Probleme rezolvate	290
5.4.1. Enunțuri	290
5.4.2. Soluții propuse	292
5.5. Probleme propuse spre rezolvare	298
5.6. Teste grilă	307
5.7. Răspunsuri la întrebările grilă	313
<b>Capitolul 6 Fișiere</b>	<b>315</b>

6.1.	Tipuri de fișiere. Operații cu fișiere	315
6.1.1.	Operații cu fișiere	315
6.1.2.	Prelucrarea fișierelor la nivel inferior	316
6.1.3.	Prelucrarea fișierelor la nivel superior	319
6.2.	Probleme rezolvate	355
6.2.1.	Enunțuri	355
6.2.2.	Soluții propuse	355
6.3.	Probleme propuse spre rezolvare	360
6.4.	Teste grilă	371
6.5.	Răspunsuri la întrebările grilă	378
<b>Capitolul 7</b>	<b>Sistemul de intrare/ieșire</b>	<b>379</b>
7.1.	Testarea și modificarea stării unui flux	385
7.2.	Formatarea datelor din fluxurile de intrare/ieșire	387
7.2.1.	Formatarea prin manipulatori	389
7.2.1.1.	Manipulatori fără parametri	390
7.2.1.2.	Manipulatori cu parametri	391
7.2.2.	Formatarea prin metode	393
7.3.	Metodele clasei istream	395
7.4.	Metodele clasei ostream	397
7.5.	Manipulatori create de utilizator	399
7.6.	Fluxuri de date pentru fișiere	400
7.7.	Fișiere binare	407
<b>Capitolul 8</b>	<b>Functii din biblioteca standard</b>	<b>411</b>
8.1.	Alocarea dinamică a memoriei	411
8.2.	Sortare și căutare	412
8.3.	Functii de clasificare	414
8.4.	Operații cu blocuri de memorie	416
8.5.	Operații cu siruri de caractere	418

8.6. Biblioteca matematică	422
8.7. Programe demonstrative	425
Capitolul 9 Teste recapitulative	429
Testul numărul 1	430
Testul numărul 2	432
Testul numărul 3	434
Testul numărul 4	436
Testul numărul 5	438
Testul numărul 6	440
Testul numărul 7	442
Testul numărul 8	444
Testul numărul 9	446
Testul numărul 10	448
Testul numărul 11	450
Testul numărul 12	453
Testul numărul 13	456
Testul numărul 14	459
Testul numărul 15	462
Testul numărul 16	464
Testul numărul 17	466
Testul numărul 18	468
Testul numărul 19	470
Testul numărul 20	472
Testul numărul 21	474
Testul numărul 22	476
Testul numărul 23	478
Testul numărul 24	480
Bibliografie	482



# Introducere

**C**alculatoroarele electronice au apărut în jurul anului 1940. Totuși unele mașini mecanice de calculat au fost propuse anterior:

- Pascal 641 - realiza numai adunare și scădere
- Leibnitz 1674 - realiza și înmulțire
- Babbage 1840 - mașină analitică

Chiar și în aceste aparate primitive pot fi găsite unele din componentele unui arhitectură modernă - de exemplu un registru temporar de memorie care menține variabilele aflate în curs de procesare. Acest registru este cunoscut sub denumirea de registru acumulator. Calculatoroarele electronice au fost inițial propuse sub forma unui aparat abstract care poate realiza sau simula orice tip de mașină mecanică. Forma teoretică a fost introdusă de Alan Turing pe la mijlocul anului 1930.

Ca și în cazul multor alte tehnologii, dezvoltarea practică a unei mașini care să respecte acest model, a fost influențată de al doilea război mondial. Acest lucru a fost posibil deoarece se dorea calcularea traiectoriilor de rachete (Collosus și Eniac) sau decodarea codului Enigma folosit de Germania pentru protecția mesajelor.

Primul program stocat electronic a fost folosit în 1948. În primii ani de dezvoltare (1950) computerele și impactul lor ulterior în viața umană au fost mult subevaluate. Este de notorietate declarația lui Bill Gates, fondatorul Microsoft cum că nu vede cum ar putea fi vreodată nevoie de mai mult de 640 Ko RAM.

În anii '70 a apărut pastila de siliciu și tehnologia VLSI (Very Large Scale Integration) ca tehnică de producție, fapt ce adus la computere ieftine și

rapide. Acest lucru a condus la creșterea și diversificarea rapidă a categoriilor de utilizatori care își puteau permite să le achiziționeze.

Perioada 1970-1980 care este caracterizată de apariția microsistemeelor pe 8 biți, care deși erau scumpe permiteau accesul separat de *main-frame* la posibilitățile sistemului de calcul. În această perioadă este demnă de remarcat apariția în 1973 a lui I8080 care poate fi considerată un punct de referință în domeniu. Din acest moment softul a început să fie dezvoltat nu numai pentru calcule extrem de complexe ci și pentru jocuri, editoare de texte, programe de baze de date și alte facilități cerute de piață.

Tot în această perioadă, forțată și de ruperea de main-frame, a pornit și dezvoltarea sistemelor de operare "locale" CPM și apoi DOS.

Sistemul de operare MS-DOS a fost realizat prin adaptarea la noile instrucțiuni mașină a unui *microkernel* de UNIX la care s-au adăugat funcțiile necesare controlului perifericelor care începuseră să fie atașate calculatoarelor personale.

Din 1980 până în 1985, deși încă foarte scumpe, au început să apară pe piață arhitecturi din ce în ce mai puternice pe 16, 32 biți, iar aria lor de folosire s-a extins foarte mult și în alte domenii.

Nu este de neglijat nici impactul produs de interfețele vizuale și de controalele prin zone active ale ecranului. Aceste controale care sunt activate din *mouse* sunt caracteristice sistemului de operare WINDOWS. Tot în această perioadă, din rațiuni economice și practice, s-a produs dispariția microcalculatoarelor acestea fiind mult depășite de cerințele pieții.

Din 1985 se poate spune că s-a intrat într-o nouă eră în acest domeniu prin ieftinirea dramatică a hardware-ului și apariția rețelelor de calculatoare.

O dată cu ieftinirea arhitecturilor de calcul s-a putut diversifica și mai mult aria de utilizare a lor. Acest lucru a avut ca rezultat o creștere a prețului softului.

Conceptul de rețea permitea transferuri rapide de date, a căror cantitate a început să fie din ce în ce mai mare.

Înțial majoritatea rețelelor de calculatoare erau "închise" în sensul că nu comunicau între ele. Apoi în ultima perioadă termenul de sistem "deschis" capătă din punct de vedere practic o răspândire din ce în ce mai mare la nivel

global. Din acest punct de vedere rețelele de tip Arpanet, Internet și.a.m.d sunt reprezentative.

În ultimii ani însuși conceptul de transmisie de informație interumană tinde să se schimbe. Televiziunea digitală este deja implementată, iar prețul perifericelor dedicate a scăzut extraordinar astfel încât se preconizează dezvoltarea calculatorului astfel încât el să nu mai fie strict orientat spre aplicații dedicate ci să fie o legătură dinamică de orice tip (audio, video, teleworking) a individului cu societatea.

Vom prezenta câteva din aplicațiile mai cunoscute ale calculatoarelor în diverse domenii:

- Industrie: control și automatizare → microcontrolere, CAM (Computer Aided Manufactory)
- Economie: gestiune, transferul banilor, "banii electronici"
- Medicină: diagnoză automată, cercetarea electronică la diverse nivele ale organismului uman, analiza codului genetic, diagnoză automată → rețele neurale, stații grafice, sisteme expert
- Pictură: noi ramuri ale picturii folosind computerul, restaurare → holografie, rețele neurale, procesoare analogice
- Muzică și televiziune: prelucrări de imagini și sunete practic de orice natură → procesoare analogice, stații grafice, arhitecturi MMX
- Proiectare : programele de tip CAD (Computer Aided Design)

Se mai pot da exemple în multe domenii, însă este cert că la ora actuală viața și societatea umană nu mai pot fi concepute fără calculator.

## Programarea și limbaje de programare

Prin **programare** se înțelege în mod generic transpunerea unor operații asupra unui set de date, într-un limbaj inteligibil de către un sistem de calcul care urmează ulterior să le execute. Acest lucru este realizat în două etape:

- ✓ etapă în care este implicat omul și anume cea de trecere de la problema reală la transpunerea într-un limbaj de programare
- ✓ o a doua etapă, automată, care transpune **codul sursă** (înșiruirea de instrucțiuni specifice limbajului respectiv) într-un **cod direct executabil**

(inteligibil sistemului de calcul) lucru de care se ocupă programe specializate numite **compilatoare**

Rolul programării este ca fiind dată o anumită operațiune sau suita de operații care se aplică asupra unor seturi de date mereu diferite, să fie scris un program care să ceară setul de date **de intrare** (cele care trebuie să fie prelucrate), să execute asupra lor suita standard de operații și să livreze **datele de ieșire** (adică rezultatele).

În aceste condiții programul trebuie să fie compus din mai multe instrucțiuni, în primul rând cele care **rezervă** (declară o zonă de memorie ca fiind special dedicată numai pentru anumite date), o zonă de memorie în care se vor citi datele de intrare, o zonă de memorie în care se vor păstra datele **intermediare** (cele care apar în decursul calculelor, dar de care utilizatorul nu are nevoie) și de asemenea pe cele de ieșire. Programul trebuie să conțină instrucțiuni care să citească de la utilizator datele de intrare, urmate în sfârșit de instrucțiunile de calcul propriu-zis, ca în final să conțină instrucțiunile de afișare a rezultatelor (datele de ieșire).

## Arhitectura unui sistem de calcul

Ca *observație* de bază: un sistem de calcul este pur și simplu versiunea mult dezvoltată a unui calculator de buzunar și deci nu trebuie să ne așteptăm de la el ca vreodată să facă altceva decât l-a pus cineva, un programator prin intermediul unui program. "Inteligenta" aparentă a unui sistem de calcul provine din cantitatea de inteligență umană investită în respectivul program aflat în execuție.

Hilar, am putea spune că omul încearcă să "facă pe cineva după chipul și asemănarea lui" dar rezultatul este o copie palidă care de altfel este extrem de utilă. Din aceste lucruri derivă vestitul principiu de aur al primilor programatori **GIGO (Garbage In Garbage Out)**, care într-o traducere prozaică înseamnă: "gunoi introduci gunoi obții" și care are meritul, în afară de cel umoristic, de a atrage atenția oricui care se va apuca de programare că din nefericire calculatorul face numai ce i se indică prin diverse metode să facă.

De aici rezultă o primă atenționare valabilă aproape tot timpul în cazul **depanării** (execuției pas cu pas în scopul găsirii eventualelor defecte în funcționare) unui program: *CAUTĂ ÎNTÂI EROAREA TA DE PROGRAMATOR ȘI APOI DĂ VINA PE CALCULATOR.*

Desigur există un procent cam de 5% din situații în care un program scris corect este executat greșit, lucru care apare ca rezultat al interferenței unor programe neautorizate, cum ar fi virușii, cu programul aflat în execuție, în cazul în care nu este valabil acest lucru problema este mai complexă putând fi implicate defecțiuni aleatorii ale părții hardware. Oricum ar fi, aceste lucruri intră în sfera de acțiune a unui inginer de sistem, și deci nu sunt luate aprioric în considerație, fiind de obicei rare.

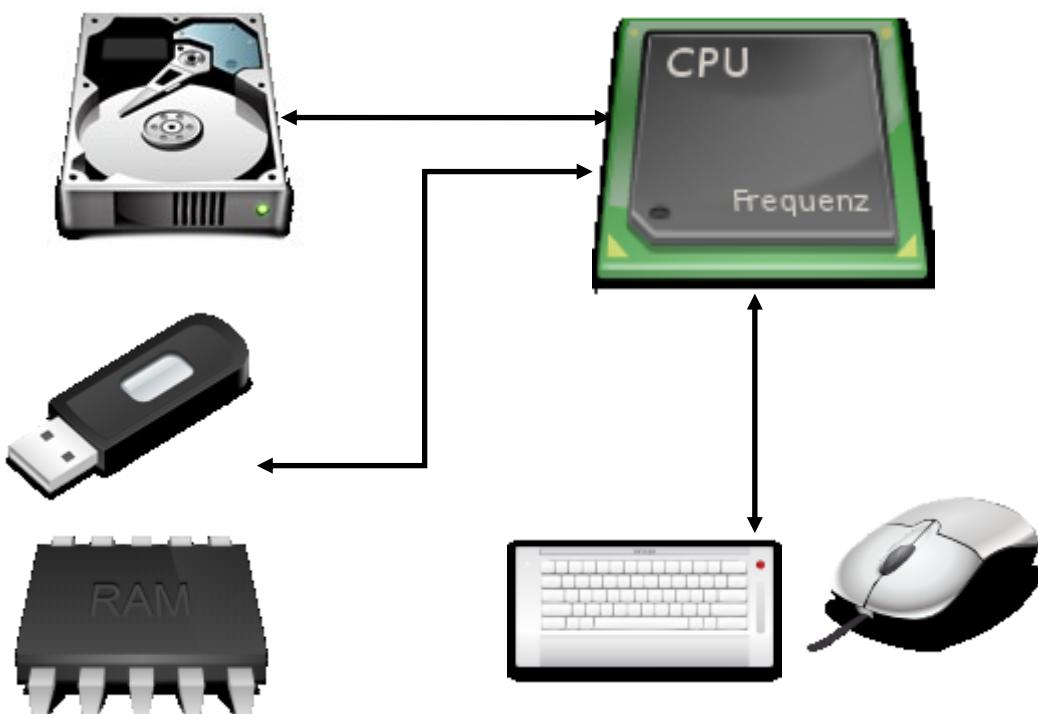
Pentru a putea înțelege arhitectura vom căuta anumite analogii între un om și un sistem de calcul.

Să analizăm ce se întâmplă cu un om în momentul în care vede o formulare de genul:  $2 \cdot 2 = \dots$ .

Deși ni se pare ceva evident, cu toții am învățat pe de rost tabla înmulțirii să încercăm acum o descompunere a pașilor care sunt execuții.

În primul rând memorăm în memoria temporară (sau de scurtă durată) **operanții** (datele asupra cărora vom executa o operație) apoi realizăm operația de înmulțire rezultând 4, care este până la comunicarea lui verbală sau prin scris tot în memoria temporară. Se observă că operanții 2,2 și operația \* au fost preluate de pe un suport extern și anume cel scris, rezultatul putând fi **returnat** (trimis către emițătorul unei cereri de lucru), după cum am spus tot pe același sistem extern de stocare.

Acest exemplu nu a fost dat fără un scop și anume, rolul lui este printre altele, de a vă obișnui cu nivelul extrem de scăzut la care se face analiza și descompunerea unei probleme reale în momentul în care dorim să o transpunem într-un limbaj de programare. Și acum să analizăm **arhitectura bloc** (structura funcțională) a unui sistem de calcul.



*Fig. 1.1 Arhitectura unui sistem de calcul*

După cum se observă din fig. 1.1 într-un sistem de calcul există:

- un dispozitiv de stocare în masă, rolul caietului pe care scriem /citim în cazul nostru. Acest dispozitiv este materializat prin Hard Disk (HDD) căruia i se mai spune și suport fix de date și Flash Drive căruia i se mai spune și suport mobil de date.
- memorie temporară, RAM (Random Acces Memory) care joacă exact rolul memoriei umane de scurtă durată.
- dispozitive de intrare sau de ieșire prin care sistemul primește sau returnează date. În cazul unui sistem de calcul acestea sunt tastatura și mouse ca dispozitive de intrare și monitorul sau imprimanta ca dispozitive de ieșire.

A mai rămas de lămurit problema microprocesorului care este "inima" unui sistem de calcul și are rolul de a coordona întregul sistem. Desigur că această coordonare se realizează tot prin intermediul unor programe care există în ceea cea am putea numi, prin analogie cu omul, partea de reflexe

condiționate. Aceste programe se găsesc într-o memorie ROM (Read Only Memory) deci nu pot fi decât citite.

## Funcționarea unui sistem de calcul

Sistemul de calcul după cum se observă comportă destul de similitudini cu omul la nivel de arhitectură și de asemenea, după cum vom discuta, și la nivel de funcționare.

Să vedem cum se realizează, bineînțeles la nivel simplificat, execuția unei formule de genul  $2*2=...$ .

Se citește de pe suportul de stocare programul care conține cererea noastră, se încarcă în memoria temporară - de acest lucru se ocupă **sistemul de operare** (programul specializat care se ocupă de traducerea unor instrucțiuni simple și intuitive ale utilizatorului în comenzi directe către partea de execuție a sistemului), în urma cererii noastre explicate de lansare a programului și se începe execuția lui de către procesor, execuție care urmează aproximativ sistemul uman de gândire. Pe scurt modul de execuție la nivelul procesorului poate fi descris ca:

- ✓ citește datele utile (din RAM în acest caz) (2,2)
- ✓ citește operația care trebuie executată asupra datelor anterioare (tot din RAM)(\*)
- ✓ execută operația ( $2*2=4$ )
- ✓ returnează rezultatul (înapoi în RAM) (4).

La nivelul sistemului de calcul modul de execuție arată ca mai jos:

- ✓ citește programul de executat (de pe sistemul de stocare fix/mobil) și
- ✓ încarcă-l în RAM
- ✓ execută programul (din RAM)
- ✓ la terminarea programului așteaptă noua cerere de execuție

## Memoria Temporară a unui sistem de calcul

După cum am pomenit în primul capitol, una din componentele de bază ale unui sistem de calcul este memoria dinamică RAM sau memoria temporară.

Această memorie a fost concepută ținând cont de faptul că sistemul de numerație folosit în interiorul unui calculator este baza 2. De aici rezultă că orice fel de dată sau instrucțiune efectiv procesată sau executată la nivelul mașinii trebuie să fie păstrată sub forma ei binară (deci corespunzătoare bazei 2). Întrucât în baza 2 nu există decât două cifre 0 și 1, rezultă că vom avea reprezentări sub forma unor șiruri de 0 și de 1. La nivelul mașinii materializarea efectivă a respectivelor numere este realizată, de obicei, prin atașarea logică a lui "1" la o tensiune de 5 volți, iar lui "0" lipsei acestei tensiuni. Totuși indiferent de nivelul tehnologic este absolut nepractic ca să considerăm fix 5 V sau fix 0 V ca valori de comparație pentru că acest lucru ar duce la niște echipamente extrem de scumpe și de lente. Din aceasta cauză se consideră a fi 0 sau 1 logic în jurul valorilor sus menționate cu  $\pm \Delta V$ . De aici a rezultat posibilitatea folosirii unui condensator ca element de memorare. Acest lucru este posibil deoarece în cazul unei reîncărcări periodice a acestuia se poate obține ca tensiunea la bornele lui să varieze în limitele necesare nouă dacă se face suficient de repede această reîncărcare. În aceste condiții foarte simplist putem vedea o memorie dinamică ca o matrice imensă de condensatoare care pot fi încărcate sau descărcate în cursul operațiilor de scriere/citire.

Aici mai apare o problemă și anume numărul de biți care compun datele utile de calcul (bit = unitate informațională corespunzătoare existenței sau a unei tensiuni la intrarea unui dispozitiv digital) și care pot fi procesați la un moment dat de un procesor.

Întrucât memoria RAM este cea care furnizează respectivele date era logic ca ea să fie organizată în diviziuni de o mărime egală cu cea a numărului de biți sus menționați. Aceste grupări se numesc **locații de memorie** și pot fi considerate ca minima unitate care poate fi scrisă sau citită dintr-o memorie. Odată lămurit acest aspect devine clar că pentru a ne putea referi la

una sau alta din locații trebuie să existe o metodă de individualizare strictă. S-a ales cea mai simplă cu puțință și anume numerotarea locațiilor - **adresa locației**.

## Noțiunea de algoritm

În procesul de rezolvare a unei probleme folosind calculatorul există două etape:

- ✓ definirea și analiza problemei
- ✓ proiectarea și implementarea unui algoritm care rezolvă problema

Definirea și analiza problemei poate fi la rândul ei descompusă în:

- ✓ specificarea datelor de intrare
- ✓ specificarea datelor de ieșire

Specificarea datelor de intrare constă în următoarele precizări:

1. Ce date vor fi primite la intrare
2. Care este formatul (forma lor de reprezentare) a datelor de intrare
3. Care sunt valorile permise sau nepermise pentru datele de intrare
4. Există unele restricții (altele decât la 3) privind valorile de intrare
5. Câte valori vor fi la intrare, sau dacă nu se poate specifica un număr fix de valori, cum se va ști când s-au terminat de introdus datele de intrare

Specificarea datelor de ieșire trebuie să țină cont de următoarele aspecte:

1. Care din valorile rezultate în cursul aplicării algoritmului de calcul, asupra datelor de intrare, vor fi afișate (necesare utilizatorului), în acest pas se face diferențierea clară între date intermedii și date de ieșire.
2. Care va fi formatul datelor de ieșire (de exemplu un număr real poate fi afișat cu trei sau cu cinci zecimale, sau un text poate fi afișat integral sau parțial)
3. Sunt sau nu necesare explicații suplimentare pentru utilizator în afara

datelor de ieșire.

4. Care este numărul de date de ieșire care trebuie transmise către ieșire.

O definiție a noțiunii de algoritm poate fi: înlățuirea de pași simpli, operații distincte care descriu modul de prelucrare a unor date de intrare în scopul rezolvării unei probleme. Deci, ALGORTIMUL ESTE ACEA PARTE DE CALCUL COMUNĂ ÎN CAZUL REZOLVĂRII UNEI PROBLEME DE CALCUL.

Un exemplu simplu de algoritm ar fi suita de operații matematice făcută în rezolvarea unei ecuații matematice de gradul al II-lea  $aX^2 + bX + c = 0$ , coeficienții  $a$ ,  $b$ ,  $c$  se schimbă, dar modul de procesare a valorilor lor nu.

Proprietățile unui algoritm sunt:

1. Este compus din instrucțiuni simple și clare.
2. Operațiunile specificate de instrucțiuni se execută într-o anumită secvență.
3. Soluția trebuie obținută într-un număr finit de pași.

Din cele spuse mai sus rezultă că UN ALGORITM ESTE INDEPENDENT DE TIPUL DE LIMBAJ ÎN CARE ESTE TRANSPUS SAU DE TIPUL DE MAȘINĂ PE CARE ESTE EXECUTAT.

Observație: în această carte se vor discuta numai ALGORITMI SECVENTIALI și respectiv PROGRAMAREA SECVENTIALĂ.

Un algoritm secvențial este acel algoritm creat pentru execuția pe un calculator secvențial. Denumirea de secvențial provine din faptul că acesta nu poate executa decât o singură operație la un moment dat.

## Reprezentarea algoritmilor

În general, un algoritm poate fi considerat ca o descriere a prelucrării efectuate asupra unui flux de date, prelucrare care are loc cu un scop bine determinat. Ca de obicei, în cazul dezvoltării unei noi teorii, s-a căutat atât o cale de standardizare a modului de descriere a unui algoritm, cât și o cale de a-l face independent de un limbaj de programare astfel rezultând două metode clasice:

1. metoda schemei logice
2. metoda pseudocodului

## 1. Metoda schemei logice

În cadrul acestei metode se folosește un set de simboluri, prezentat în figura 1.2, pentru descrierea pașilor ce trebuie executați pentru ca programul rezultat să ne rezolve o anumită problemă.

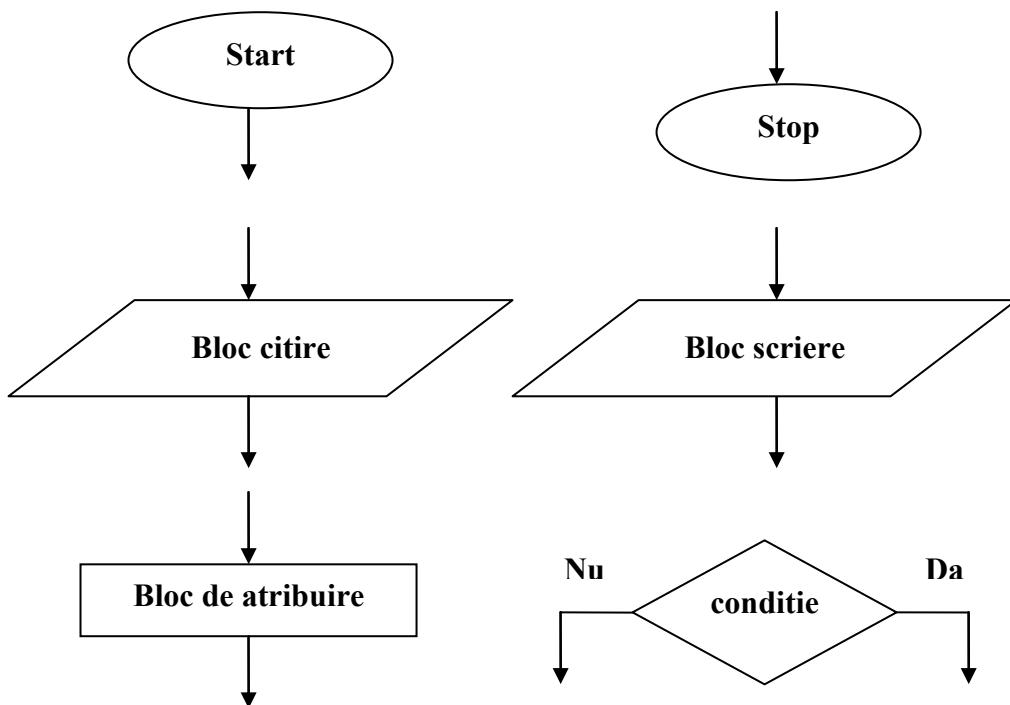
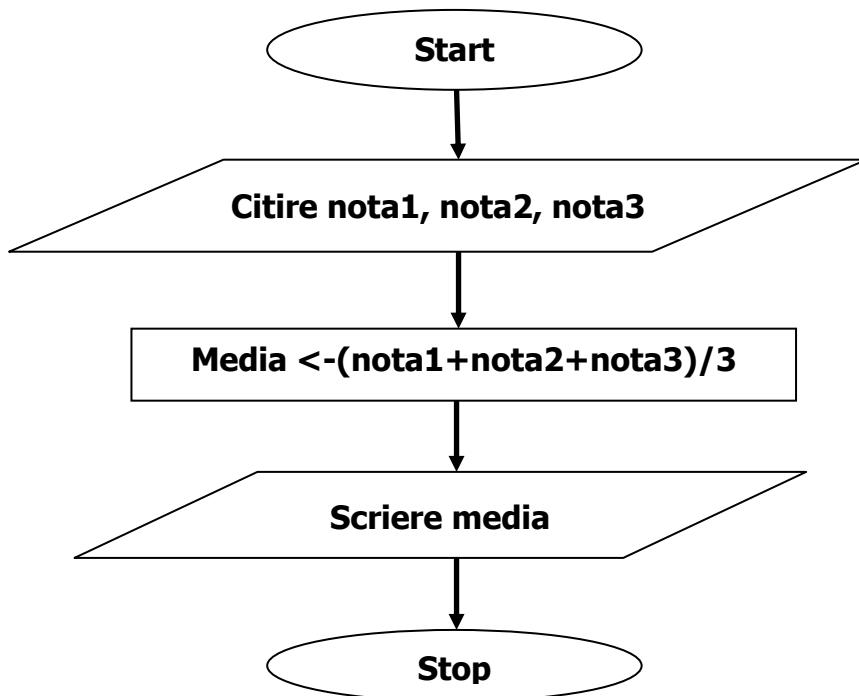


Fig. 1.2 Simbolurile utilizate în pseudocod

Deși a fost extrem de folosită, până nu de mult, această metodă pare a pierde teren în fața reprezentării de tip pseudocod, poate și datorită timpului suplimentar pierdut de utilizator cu executarea simbolurilor grafice.

Să analizăm un program de calcul a mediei pentru trei note și să vedem cum ar apărea descris prin această metodă.



*Fig. 1.3 Exemplu de algoritm scris cu ajutorul pseudocod-ului*

După cum se observă în figura 1.3 se descriu efectiv pas cu pas toate operațiile care trebuie efectuate.

În figura 1.3 s-a specificat și începutul programului, fapt care pare inutil, dar în cazul descrierii unui program foarte mare se prea poate ca schema logică să se întindă pe zeci de pagini, caz în care devine evidentă necesitatea declarării începutului și sfârșitului de program.

În următorul pas se declară variabilele. O variabilă este un identificator propriu al programatorului, care în cazul operațiunii de declarare se atașează, la generarea programului, unei zone de memorie a cărei dimensiune este în directă concordanță cu tipul de dată stocat de aceasta.

Se trece apoi la citirea datelor de intrare, în acest caz este vorba de trei note ceea ce presupune că valorile de intrare pot avea și zecimale. Mai târziu se va discuta despre alegerea tipului de variabilă în funcție de specificațiile utilizatorului.

Se execută calculul propriu zis, după care se afișează rezultatul cu 2 zecimale. A mai rămas doar marcarea sfârșitul de program. Chiar dacă metoda are o impresie "artistică" sporită câteodată se pierde prea mult timp și

hârtie în cazul folosirii ei și de aceea din ce în ce mai mulți preferă ca metodă de descriere pseudocodul.

## 2. Metoda pseudocodului

Față de schema logică, are dezavantajul că nu întotdeauna redă absolut toți pașii de calcul. Totuși este mai comodă de folosit în cazul în care se descriu algoritmi pentru limbajele de programare de nivel mediu și înalt.

Schema logică este cea mai indicată ca metodă de descriere, în special, în cazul implementării unui algoritm folosind limbaj de asamblare, unde descrierea amănunțită a fiecărui pas este necesară pentru putea minimiza şansele apariției unor erori de implementare.

Descrierea efectivă a pașilor ce trebuie efectuați, folosind cuvintele limbii române împreună cu cuvinte cheie ale limbajului de programare în care se va implementa algoritmul se numește pseudocod. Se observă că în acest caz criteriul independenței totale a descrierii față de limbajul de implementare nu este întru totul respectat.

Pentru comparație vom descrie mai jos în pseudocod problema prezentată în figura 1.3.

```
variabile: med_a,n1,n2,n3
{
    citește n1, n2, n3
    med_a = (n1 + n2 + n3) / 3
    afișează med_a cu 2 zecimale
}
```

## Limbajul C

Limbajele de programare de nivel mediu au fost serios dezvoltate pe la mijlocul anilor '50. Ideile de bază folosite pentru proiectarea lor derivă din formulele funcționale X-Church ce fuseseră propuse prin anii '30. La ora actuală se estimează că există peste 2000 de limbi de programare, diferențele între ele fiind legate în principal de stilul de programare.

Limbajul FORTRAN (FORmula TRANslator) a fost unul dintre primele limbi de programare. El este folosit și acum în industrie sau cercetare pentru a implementa aplicații specifice cu volume mari de calcule matematice complexe.

Pe lângă limbi de programare pentru mașini secvențiale, s-au dezvoltat și limbi care suportă procesare paralelă. Din nefericire majoritatea nu sunt portabile, fiind orientate strict pe arhitectura mașinii pentru care a fost dezvoltat limbajul. De abia în ultimii zece ani dezvoltarea rapidă a rețelelor de calculatoare a condus și la o încercare de standardizare a unor limbi ce pot crea cod executabil simultan pe mai multe sisteme de calcul. Avantajele programării la nivel înalt sunt:

- ascund diferențele de arhitectură internă
- sunt ușor de urmărit și menținut (modificare și depanare ușoară)
- sunt portabile. De exemplu un cod scris în C poate fi executat și sub UNIX și sub DOS
- nu au cel mai eficient cod, în comparație cu cel scris în ASM direct

De obicei sistemele de operare erau dezvoltate în limbaj de ansamblare pentru a fi mai rapide. Limbajul C, dezvoltat în 1972 de Dennis M. Retchie la Laboratoarele AT&T Bell, este primul limbaj pentru crearea sisteme de operare. Numele limbajului provine din faptul că este rezultatul îmbunătățirii limbajului B, folosit în scrierea UNIX pentru DEC PDP7. Prima documentație despre acest limbaj a fost "The C Programming Language", scrisă de Retchie și Brian Kernighan în 1977. Înaintea ei există doar "The C Reference Manual", scrisă de Retchie. O caracteristică importantă a acestui limbaj este faptul că

poate fi considerat simultan și un limbaj de nivel mediu și un limbaj de nivel scăzut.

O dată cu răspândirea lui printre programatorii profesioniști s-a trecut și la adaptarea compilatorului pentru sistemul de operare MS-DOS. De asemenea a apărut și altă problemă și anume faptul că standardul K&R nu mai era satisfăcător. De asemenea datorită variantelor de compilare pentru PC-uri, limbajul C începuse să-și piardă din portabilitate.

În acel moment Institutul Național pentru Standarde al Americii (ANSI), a creat un subcomitet pentru a defini o versiune oficială pentru limbajul C, rezultând astfel ANSI C.

Limbajul C și versiunile sale OOP (Object Oriented Programming) C++ și mai nou Visual C++ sunt printre cele mai folosite limbaje de programare la ora actuală. A fost proiectat pornind de la două limbaje dezvoltate spre sfârșitul anilor '60: BCPL și prima sa versiune B.

Un limbaj de nivel scăzut este orientat pe arhitectură și poate fi definit ca având facilitatea de a scrie instrucțiuni ce se referă direct la elemente ale respectivei arhitecturi interne (ca în cazul limbajului de asamblare).

Un limbaj de nivel înalt sau mediu este orientat pe problemă permitând programatorului să-și structureze programul cât mai apropiat posibil de problema ce trebuie implementată. Acest lucru scade șansele unei erori de proiectare, la nivel logic, a aplicației. Chiar dacă programul va fi mai ineficient din punct de vedere al vitezei și al ocupării resurselor (prin comparație cu un program scris în limbaj de asamblare) el are avantajul că poate fi scris repede și extins cu ușurință.

Limbajul C permite folosirea ambelor tehnici: programare structurată și acces direct la mașină, fapt care-l face să fie foarte flexibil.

Ultimul și poate cel mai important motiv pentru învățarea limbajului C este faptul că permite trecerea cu ușurință la C++ sau Java.

Deși limbajul C are o libertate foarte mare în scrierea codului, el acceptând multe forme de scriere care în alte limbaje nu sunt premise, acest lucru ascunde însă și pericole, în special pentru programatorii fără experiență.

## Structura generică a unui program scris în C

De la primele programe scrise este bine să se țină cont de o serie de reguli privind organizarea codului sursă. Structura generală a unui cod va fi:



*Apeluri ale directivelor speciale de compilare;  
Declarații de biblioteci folosite;  
Declarații funcții in-line;  
Declarații ale prototipurilor funcțiilor personale;  
Declarații variabile globale;  
Corpul programului principal  
{  
declarații variabile;  
apeluri de funcții;  
}  
Corpurile funcțiilor dezvoltate de programator;*

Există câteva observații suplimentare privind tehnoredactarea codului sursă de care ar fi bine să se țină seama.

1. Să nu se scrie instrucțiunile una după alta ci una sub alta.
2. Toate instrucțiunile care formează un bloc pentru o altă instrucțiune să fie deplasate față de începutul respectivei instrucțiuni.

## Etapele generării unui program executabil pornind de la o problemă reală

Una din greșelile începătorilor în programare constă în faptul că pun prea mult accent pe introducerea unor programe gata scrise, fără ca în prealabil să studieze problema de pornire precum și algoritmul generat în urma analizei acestei probleme. În momentul gândirii unui program, pe lângă alte considerente trebuie ținut cont și de faptul că este o mare diferență între programele bune și cele care merg. Un program bun nu numai că merge, dar este și ușor de înțeles și menținut. Mulți dintre începătorii în programarea C-ului au tendința de a se limita la programe care să meargă, și acest fapt conduce la deprinderi nesănătoase.

De aceea mai jos vom prezenta pașii necesari a fi urmați în elaborarea unei aplicații pornind de la o problemă reală.

1. Înțelegerea problemei.
2. Realizarea modelului matematic, dacă este cazul.
3. Verificarea modelului matematic cu un set de date reale sau calculate manual. Dacă se observă neconcordanțe se va relua verificarea punctelor anterioare dacă nu, se continuă.
4. Identificarea datelor de intrare, a datelor de ieșire precum și a variabilelor intermediare.
5. Alegerea tipurilor de date folosite pentru reținerea în cursul procesării a respectivelor date.
6. Specificarea modului de interfațare cu utilizatorul la nivel de intrare sau ieșire.
7. Generarea schemei logice sau a pseudocodului ce descrie algoritmul propus pentru rezolvarea problemei.
8. Verificarea modelului creat la punctul 7 cu un set date reale sau calculate manual. În caz de neconcordanțe se reia verificarea punctelor anterioare dacă nu, se continuă.
9. Scrierea codului sursă (transpunerea într-un limbaj de programare oarecare).
10. Eliminarea erorilor de sintaxă.

Pentru a exemplifica pașii menționați anterior, vom lua un exemplu extrem de simplu. "Să se realizeze un program care rezolvă o ecuație oarecare de gradul al II-lea"

## Pasul 1

În acest caz este simplu, e clar că ne referim la o ecuație de forma

$$ax^2 +bx+c=0.$$

## Pasul 2

Modelul matematic este cel studiat în liceu:

Fie  $\Delta = b^2 - 4 \cdot a \cdot c$ , atunci:

$$\text{Dacă } \Delta > 0 \text{ atunci: } x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$$

$$\text{Daca } \Delta < 0 \text{ atunci: } x_{1,2} = \frac{-b \pm i \cdot \sqrt{-b^2 + 4 \cdot a \cdot c}}{2 \cdot a}$$

$$\text{Daca } \Delta = 0 \text{ atunci: } x_{1,2} = -\frac{b}{2 \cdot a}$$

### Pasul 3

În acest caz poate fi evitat modelul matematic este prea simplu.

### Pasul 4

Să vedem deci care ar fi datele ce trebuie furnizate de utilizator. Având în vedere că se dorește rezolvarea unei ecuații generice de ordinul II este clar că: a,b,c date de intrare

$X_{1,2}$  date de ieșire

$\Delta$  dată intermediară.

Analizând modelul matematic observăm că a trebuit să folosim un simbol suplimentar în funcție de care se face întreaga discuție. Acest simbol nu este nici dat de utilizator și nici nu interesează pentru rezultatul final, deci  $\Delta$  este o dată intermediară de calcul.

### Pasul 5

Acum trebuie să specificăm tipul de dată folosit pentru reprezentarea internă (în memoria calculatorului) a variabilelor. Deci avem câteva date suplimentare la dispoziție pentru a vedea tipul de reprezentare care trebuie ales:

- specificațiile utilizatorului
- modelul matematic
- experiența în programare

În acest caz modelul matematic ne spune că a,b,c sunt numere reale. Fără o specificare clară a utilizatorului care să indice că ele aparțin altui interval este obligatoriu ca să luăm în considerare modelul cel mai extins, deci cel mai adaptabil. Desigur că nu vom alege cel mai cuprindător tip de date pus la dispoziție de către compilator în lipsa unor specificații externe. Deci a,b,c vor fi de tipul **float**.

Variabila intermediară  $\Delta$  este rezultat al aplicării unei funcții lineare asupra unor variabile de tip real deci și delta va trebui să fie **float**.

Datele de ieșire  $x_{1,2}$ , după cum se observă sunt rezultatul unei împărțiri și deci obligatoriu sunt reale indiferent de tipul de dată ales pentru datele de intrare.

### Pasul 6

În lipsa unor specificații de la utilizator vom alege pentru citirea datelor cea mai simplă metodă citirea directă după cum am făcut și până acum. Pentru afișare însă avem două metode afișarea de tipul  $\text{Re}(x)$  și  $\text{Im}(x)$  sau direct rezultatul ca în calculul manual. Vom alege cea de a doua metodă.

### Pasul 7

Pseudocodul va arăta ca mai jos.

*citește a*

*citește b*

*citește c*

*calculează delta după formulă  $\delta=b^2-4*a*c;$*

*dacă  $\delta \geq 0$  atunci*

*{*

*$x_1=(-b+\sqrt{\delta})/(2*a);$*

*$x_2=(-b-\sqrt{\delta})/(2*a);$*

*afișează  $x_1, x_2$*

*}*

*altfel*

*{*

*$\delta=-\delta;$*

*$\delta=\sqrt{\delta}/(2 * a)$*

*afișează  $x_1$  de forma  $re+i*im$*

*afișează  $x_2$  de forma  $re-i*im$*

*}*

Câteva observații relative la construcția pseudocodului

- se observă că nu am tratat separat situația  $\delta=0$  pentru că zero la adunare nu modifică rezultatul
- în a doua parte am folosit un artificiu de calcul în sensul că am păstrat tot în variabila  $\delta$  rezultate parțiale de calcul. Altfel ar fi trebuit să introducem variabile suplimentare pentru păstrarea părții întregi și a

părții reale. Codul ar fi fost mai clar, dar în același timp am fi folosit mai multă memorie. Dacă în cazul acestui program câteva variabile suplimentare nu ne deranjează, în cazul unor programe foarte mari acest lucru poate ridica greutăți ce nu pot fi eliminate decât prin trecerea la alocarea dinamică a memoriei. Această metodă va fi prezentată ulterior.

- se observă că am scris mult cod suplimentar care din punct de vedere al calculului brut nu ne trebuie, dar din punct de vedere al modului în care îi prezentăm rezultatul unui utilizator este esențial pentru că este exact în forma matematică cunoscută.

### Pasul 8

Va fi evitat pentru acest exemplu deoarece este prea simplu.

### Pasul 9

Codul în cazul limbajului C va arăta ca mai jos:



```
#include <stdio.h>
#include <conio.h> // aici eroare de sintaxă detectabilă de
                   // interpretor
void main(void)
{
    float a,b,c,delta,x1,x2;
    clrscr()
    printf(" Pentru calculul ecuației a*x*x+b*x*c=0 dați
           coeficienții");
    printf("\na=");
    **      scanf("%f",a); // aici eroare de sintaxă nedetectabilă de
                           // interpretor
    printf("\nb=");
    scanf("%f",&b);
    printf("\nb=");
    scanf("%f",&b);
    ***   delta=b*b+4*a*c; //aici eroare de calcul
    if( delta >=0 )
    {
        xi=(-b+sqrt(delta))/(2 *a);
        x2=(-b-sqrt(delta))/(2 *a);
        printf("\nx1=%f",x1)
        printf("\nx2=%f",x2)
    }
    delta=-delta; delta=sqrt(delta)/(2 *a)
    printf( "x1 = %f/%f, -b/(2 *a), delta);
    printf( "x2 = %f/%f, -b/(2 *a), delta);
}
```

## Pasul 10

După cum se observă am introdus intenționat trei tipuri de greșeli. Prima, marcată cu o stea este greșeală de sintaxă ce va fi detectată de compilator și deci o putem corecta în această fază.

## Pasul 11

În cadrul rulării programului rezultat se observă că acesta nu funcționează corect. La execuția pas cu pas se va observa că în variabila a nu se depune ceea ce se citește de la tastatură. Este o greșeală de logică de programare, pentru că `scanf` așteaptă un pointer pentru a returna corect rezultatul, iar a nu este declarat pointer trebuie transmisă adresa lui a. Această eroare nu poate fi detectată de interpreter.

De asemenea cu trei stele am marcat o greșeală de transpunere a algoritmului, o greșeală simplă de transcriere a unei formule de calcul. De abia după eliminarea acestor erori se poate continua testarea programului.

Este bine ca să existe un comentariu la începutul oricărei surse C având următorul conținut:

```
/*
----- autor:
----- data creării/modificării
----- copyright
*/
```



## Elemente de bază ale limbajului C++

### 1.1. Tipuri de date. Variabile. Constante

#### 1.1.1. Cuvinte cheie

Limbajul C, ca orice limbaj de programare, este compus din câteva denumiri (identificatori) cu o semnificație bine stabilită, numite *cuvinte cheie*:

Observație: Când alegeți denumiri de variabile pentru programe să nu utilizați aceste denumiri.



<i>auto</i>	<i>break</i>	<i>case</i>	<i>char</i>	<i>const</i>	<i>continue</i>
<i>double</i>	<i>else</i>	<i>enum</i>	<i>extern</i>	<i>float</i>	<i>for</i>
<i>int</i>	<i>long</i>	<i>register</i>	<i>return</i>	<i>short</i>	<i>signed</i>
<i>struct</i>	<i>switch</i>	<i>typedef</i>	<i>union</i>	<i>unsigned</i>	<i>void</i>
<i>default</i>	<i>do</i>				
<i>goto</i>	<i>if</i>				
<i>sizeof</i>	<i>static</i>				
<i>volatile</i>	<i>while</i>				

Limbajul C++ adaugă noi cuvinte cheie la cele existente ale limbajului C.



<i>asm</i>	<i>bool</i>	<i>catch</i>	<i>class</i>	<i>delete</i>	<i>friend</i>	<i>inline</i>
<i>namespace</i>	<i>new</i>	<i>operator</i>	<i>private</i>	<i>public</i>	<i>protected</i>	<i>plate</i>
<i>using</i>	<i>virtual</i>					

### 1.1.2. Tipuri de date

Un **tip de date** specifică mulțimea de valori pe care variabila respectivă le poate lua cât și setul de operații pe care programatorul le poate efectua cu acea variabilă.



#### NUMELE TIPULUI

**char**

**int**

**float**

**double**

**void**

#### CARACTERISTICI

retine un singur caracter; Ex: 'A', 'a', '%', etc.

retine numere intregi cu semn;  
Ex: 23,-45,0, etc.

retine numere reale în format cu virgula mobila, în simpla precizie;

Ex: 7.8965, -4.123, 7.0, etc.

retine numere reale în format cu virgula mobila, în dubla precizie;

Ex: 7.8965, -4.123, 7.0, etc.(se utilizeaza cand se prelucreaza numere foarte mari sau foarte mici)

tip de date special care nu specifică un anumit set de valori initial, dar care poate fi specificat ulterior declararii.

#### Tipul de date char



**char <definitie\_de\_data>;**

Se reprezintă în memoria calculatorului folosind 8 biți (un octet) și poate păstra valori cuprinse între -128 și 127. Programatorii pot atribui valori de tip caracter unei astfel de variabile în două modalități distincte, dar care acționează identic. Astfel se poate folosi reprezentarea din ASCII (codul numeric al caracterului respectiv), sau caracterul respectiv între două apostrofuri. Dacă se declară fără semn (adică se utilizează modificadorul *unsigned*), intervalul de valori se întinde de la 0 la 255.



Exemplu:      **char litera\_mica;**  
                  **litera\_mica = 90;**  
                  sau **litera\_mica = 'a';**

## Tipul de date int



```
int <definitie_de_data>;
```

Se reprezintă în memoria calculatorului folosind 16 biți (2 octeți) și poate păstra valori cuprinse între -32768 și 32767. Dacă se declară fără semn (adică se utilizează modifierul *unsigned*), intervalul de valori se întinde de la 0 la 65535.



Exemplu:

```
int a = 9;  
int b = 4567;  
int c = -31456;
```

## Tipul de date float



```
float <definitie_de_data>;
```

Se reprezintă în memoria calculatorului folosind 32 biți (4 octeți) și poate păstra valori cuprinse între 3.4E-38 și 3.4E+38.



Exemplu:

```
float x = 9.789;  
float y = -6754.123;  
float z = -3124567;
```

## Tipul de date double



```
double <definitie_de_data>;
```

Se reprezintă în memoria calculatorului folosind 64 biți (8 octeți) și poate păstra valori cuprinse între 1.7E-308 și 1.7E+308.



Exemplu:

```
double numar_foarte_mare=12345678912345.123456789123456789;  
double numar_foarte_mic=-123456789123456789.12345678912345;  
double numar_mare=-123456789;
```

## Tipul de date void



[void ] <definitie\_de\_functie([void])  
sau  
void <definitie\_de\_pointer>;

Este tipul de dată vidă (fără tip specificat), utilizat în general pentru mărirea clarității programelor. Tipul void permite explicitarea faptului că o funcție nu returnează nimic sau nu are nici un parametru.



Exemplu:

```
void salut(void)  
{  
    cout<<“SALUTAM PROGRAMATORII IN LIMBAJUL C++!!!\n“;  
}
```

## Modificatorii de tip

Limbajul C++ oferă pe lîngă cele 5 tipuri de bază prezentate mai sus, un set de modificatori de tip : *unsigned* (fără semn), *long* (lung), *signed* (cu semn) , *register* (registru) și *short* (scurt).

Un *modificator de tip* schimbă domeniul valorilor pe care o variabilă le poate păstra, sau modul în care compilatorul păstrează o variabilă. Pentru a se modifica un tip se va plasa modificatorul în fața tipului respectiv.



Exemplu:

```
unsigned int numar;  
register int i;  
long int numar_foarte_mare;
```

Modifierul de tip *unsigned* indică compilatorului să nu folosească cel mai semnificativ bit de semn, ci să-l folosească pentru a reprezenta valori mai mari.

Astfel: *unsigned int* se folosește pentru reprezentarea variabilelor din intervalul [0,65535]; *unsigned char* se folosește pentru reprezentarea variabilelor din intervalul [0,255], etc.

Modifierul de tip *long* indică compilatorului să folosească 32 de biți (4 octeți) pentru a reprezenta un număr întreg. Variabila de tip *long* poate să păstreze valori în intervalul [-2147483648, 2147483647].

Utilizând combinat modificadorii *unsigned* și *long* se poate indica astfel compilatorului să aloce 32 de biți pentru variabilele cuprinse în intervalul [0,4292967265].

Modifierul de tip *register* se folosește când se dorește ca variabila respectivă să fie păstrată într-un registru și astfel accesul să fie mai rapid. De obicei se reține o variabilă de ciclare (contor), pe care programul o accesează la fiecare repetare a unei bucle.

### 1.1.3. Constante

Sunt date a căror valoare nu poate fi modificată în timpul execuției programului. Ele reprezintă un tip și o valoare și astfel pot fi de mai multe tipuri:

1. **constantă întreagă** = se reprezintă sub forma unei însiruiri de cifre și se clasifică în :
2. **constante zecimale** (se scriu în baza 10) Exemplu: 14, 568, 17342
3. **constante octale** (se scriu în baza 8) Exemplu: 0sir de cifre în baza 8
4. **constante hexazecimale** (se scriu în baza 16) Exemplu: 0x sir de cifre în baza 16

Constanțele întregi se reprezintă pe 16 biți sau pe 32 de biți. Dacă la sfârșitul unei constante punem litera l sau L, atunci constanta respectivă va fi reprezentată pe 32 de biți.

Exemplu: numărul 17 se reprezintă pe 16 biți

numărul 17L se reprezintă pe 32 biți

5. **constantă flotantă** este compusă din 2 părți – *partea fracționară* (care poate fi vidă) și *exponent* (care poate fi el vid)

O constantă reală este sub următoarea formă:

**parte întreagă.parte fracționară e ± exponent**

Exemplu: 3.45e-17  $\leftrightarrow$  3,45<sup>-17</sup>

Toate constantele flotante se reprezintă pe 16 biți.

6. **constantă caracter** este de fapt un caracter între apostrofuri.

Se reprezintă pe 8 biți, fiind chiar reprezentarea în codul ASCII a caracterului respectiv.

Exemplu: 'A' reprezentare internă : 65 (codul ASCII a caracterului 'A')

'a' reprezentare internă : 97 (codul ASCII a caracterului 'a')

În plus avem o notație specială '\ = backslash, care se poate folosi împreună cu câteva litere mici cu următoarele semnificații:

CARACTER	SEMNIFICATIE
\a	caracter ASCII de atenționare (sunet)
\b	backspace
\f	avans hartie
\n	linie nouă
\r	retur de car
\t	tabulator orizontal
\v	tabulator vertical
\\\	backslash
\'	apostrof
\"	ghilimele
\?	semnul întrebării

## 7. constantă sir sau sir de caractere

Acest tip de constantă apare ca o succesiune de caractere scrise între ghilimele. Poate fi și sirul vid. Reprezentarea internă este astfel încât fiecare caracter apare pe câte un singur octet, iar ca terminator de sir avem caracterul 0 (nul). Constantele sir pot și scrise pe linii diferite, dar pe prima linie ultimul caracter este backslash, înainte de apăsarea tastei RETURN.

Exemplu: linia 1 : “cont\

linia 2 : nuare”

Exemplu : “AbbA” se reprezintă intern astfel:

65	98	98	65	0
A	b	b	A	

### 1.1.4. Variabile

Pentru a putea utiliza informațiile ce pot fi prelucrate prin intermediul programelor, trebuie să folosim denumiri (identificatori), care să fie compuși din caractere – litere, cifre și liniuță de subliniere din maxim 31 caractere.

Numim **variabilă** o denumire (identificator) pe care compilatorul o asociază cu o anumită zonă de memorie.

Când se declară o variabilă, trebuie specificat numele ei cât și tipul de date asociat.



Exemplu:

```
int variabila_de_tip_intreg;  
// am declarat o variabilă de tip intreg careia i-am dat denumirea  
// variabila_de_tip_intreg  
float variabila_de_tip_real;  
// am declarat o variabilă de tip real careia i-am dat denumirea  
// variabila_de_tip_real  
char variabila_de_tip_caracter;  
// am declarat o variabilă de tip caracter careia i-am dat denumirea  
// variabila_de_tip_caracter
```

*Restricție :* Numele variabilelor nu pot să înceapă cu o cifră.

Exemplu: variabila1 – este corect  
1variabila - nu este corect

Observație: Limbajul C este *case sensitive*, adică face diferență dintre literele mici și mari, astfel încât, două denumiri de variabile sau de funcții, care sunt identice dar sunt scrise o dată cu litere mici iar apoi cu litere mari, se consideră ca fiind două denumiri de variabile sau funcții diferite.

Exemplu: int var\_intreaga;

int VAR\_INTREAGA; semnifică două denumiri total diferite.

Variabilele pot fi:

1. simple
2. compuse: tablou  
structură etc.

## 1) Variabilele simple

Declarația de variabilă simplă are forma:



**tip nume\_variabila;**



Exemplu:

```
int i;  
int j,k,l;  
double a,b;  
float x,y;  
char m,n,t;
```

## 2) Variabilele tablou

Prin tablou înțelegem o mulțime ordonată de același tip; accesul la elementele tabloului făcându-se cu ajutorul indicilor.

Declarația este:



**tip nume\_tablou[][];**



Exemplu:

```
int v[5];
float x[25];
double a[3][2];
```

Observație:

Numerotarea elementelor unui tablou în limbajul C++ începe cu indicele 0. Elementele lui int v[5] vor fi : v[0],v[1],v[2],v[3],v[4];

Indice poate să fie orice expresie întreagă. Putem avea chiar și tablouri de șiruri de caractere: char t[20];

Numele tabloului este de fapt adresa primului său element.

### 3) Inițializarea variabilelor

Poate fi făcută chiar pe linia de declarare a variabilelor:



Exemplu:

```
int i = 5;
float x = 7.8;
int v[5] = {1,2,7,10,-5};
float y[3] = {-9.034,89,2};
char c = 'B';
```

Pentru inițializarea variabilelor de tip șir de caractere avem următoarele posibilități:

```
char t[15]={‘s’,‘i’,‘r’,‘ ‘,‘c’,‘o’,‘r’,‘e’,‘c’,‘t’,‘\0’};
```

```
char t[15]=“sir corect”;
```

### 4) Comentarii în programe

Numim comentarii texte care nu sunt luate în considerare de compilatorul și care apar între simbolurile /\* comentariu \*/, sau când este vorba despre o singură linie // comentariu.

Se mai pot pune comentarii pentru ca să se eliminate una sau mai multe instrucțiuni din programul C/C++.

## 1.2. Operatorii limbajului C++

### Expresii

O expresie poate să fie un operand sau mai mulți operanzi legați prin operatori.

Orice expresie are tip și valoare care sunt date după evaluarea expresiei.

### Operatori

Operatorii folosiți în limbajul C++ au o asociere de la stânga la dreapta – în general – cu excepția operatorilor unari (se aplică la un singur operand), relaționali și de atribuire, la care asocierea se face de la dreapta la stânga. Operatorii sunt împărțiți în 11 categorii:



OPERATORI	
1	aritmetici
2	relaționali
3	de egalitate
4	logici
5	logici pe biți
6	de atribuire
7	de incrementare și decrementare
8	de conversie explicită (cast)
9	de lungime (sizeof)
10	condițional
11	virgulă

## 1. Operatori aritmetici



OPERATOR	FUNCTIE
+	adunare
-	scădere
*	înmulțire
/	împărțire
%	restul împărțirii
+	adunare unară

În cele mai simple programe se pot utiliza operații matematice cum ar fi adunarea, scăderea, înmulțirea și împărțirea.



Exemplu:

```
int i=9, j=2;  
atunci i / j are ca rezultat 4  
i % j are ca rezultat 1
```

Prezentăm în următorul program scris în C++, principali operatori matematici:



```
#include <iostream.h>  
int main(void)  
{  
    int secunde_pe_ora;  
    float media;  
    secunde_pe_ora= 60 * 60;  
    media = (5 + 10 + 15 + 20) / 4;  
    cout<<"Numarul de secunde intr-o ora este"  
    "<<secunde_pe_ora<<endl;  
    cout<<"Media numerelor 5, 10, 15 si 20 este "<<media<<endl;  
    cout<<"Numarul de secunde in 48 de minute este"  
    "<<secunde_pe_ora - 12 * 60<<endl;  
}
```

După execuția programului se vor afișa pe ecran următoarele rezultate:

*Numarul de secunde intr-o ora este 3600*

*Media numerelor 5, 10, 15 si 20 este 12.000000*

*Numarul de secunde in 48 de minute este 2880*

## 2. Operatori relaționali



OPERATOR	FUNCTIE
<	mai mic
<=	mai mic sau egal
>	mai mare
>=	mai mare sau egal

În programe, prin aplicarea acestor operatori relaționali se pot obține două valori posibile, la evaluarea expresiilor care îi conțin:

0 – ceea ce înseamnă că expresia este falsă

1 – ceea ce înseamnă că expresia este adevărată



Exemplu: int i=3, j=8;

Atunci pentru expresia  $i < j$  avem valoarea 1

Iar pentru expresia  $i >= j$  avem valoarea 0

## 3. Operatori de egalitate



OPERATOR	FUNCTIE
==	egal
!=	diferit

În programe, prin aplicarea acestor operatori de egalitate se pot obține două valori posibile, la evaluarea expresiilor care îi conțin:

0 – ceea ce înseamnă că expresia este falsă

1 – ceea ce înseamnă că expresia este adevărată



Exemplu: int i=2, j=5, k=2;

Atunci pentru expresia  $i != j$  avem valoarea 1

Pentru expresia  $i == j$  avem valoarea 0

Iar pentru expresia  $i == k$  avem valoarea 0

## 4. Operatori logici



OPERATOR	FUNCTIE
!	negare
&&	și logic
	sau logic

*În limbajul C/C++ nu există valori speciale pentru adevărat sau fals și de aceea valoarea de fals este reprezentată prin zero ( 0 ), iar valoarea de adevărat este reprezentată prin orice număr diferit de zero ( ≠0 ).*

Observație : Din punct de vedere al priorității, operatorii unari au cea mai mare prioritate și de acea, în acest caz, operatorul ! are prioritatea cea mai mare, urmat în ordine de operatorul && și de operatorul ||.

Reamintim tabele de adevăr ale celor trei operatori logici:

	<b>!0</b>	1
	<b>!1</b>	0

	<b>&amp;&amp;</b>	<b>0</b>	<b>≠0</b>
	<b>0</b>	0	0
	<b>≠0</b>	0	1

	<b>  </b>	<b>0</b>	<b>≠0</b>
	<b>0</b>	0	1
	<b>≠0</b>	1	1

Exemplu: expresia !0 are valoarea 1

expresia !5 are valoarea 0

expresia !(a==b) are valoarea 1 sau 0 în funcție de valorile lui a, respectiv b.

Prezentăm mai jos, un exemplu mai amplu cu câteva expresii la care le evaluăm valorile:



Presupunem că avem variabilele întregi  $i=0$ ,  $j\neq 0$  și  $k\neq 0$

Expresie	Valoare
$i \&& j$	0
$!i \&& j$	1
$i    j$	1
$i     j$	0
$!(i \&& j)$	1
$!i    !j$	1
$!i \&& j    !j \&& i$	1
$i <= j    j <= k$	0 sau 1

Observație: la penultima expresie  $!i \&& k || !j \&& i$ , evaluarea se face prin *scurtcircuitare*, adică o expresie logică – în C – se evaluează până în punctul în care este cunoscută valoarea expresiei. Astfel, pornim evaluarea de la stânga spre dreapta, fără a uita că operatorul unar  $!$  are prioritate mare, și avem  $!i$  adică 0 și logic cu  $j$  ( $\neq 0$ ) obținem valoarea 1, iar din tabela de adevăr a operatorului  $||$  observăm că ceva  $\neq 0$  sau logic orice valoare se obține valoarea de adevăr 1 și deci nu ne mai interesează evaluarea restului expresiei (adică  $!j \&& i$ ). Expresia la final are valoarea de adevăr 1.

## 5. Operatori logici pe biți



OPERATOR	FUNCTIE
$\sim$	negatie logică pe biți
$\&$	și pe biți
$ $	sau pe biți
$^$	sau exclusiv pe biți
$<<$	deplasare la stânga pe biți
$>>$	deplasare la dreapta pe biți

Operatorii logici pe biți se pot folosi în limbajul C, pentru a mări performanțele unui program sau pentru a-i reduce consumul de memorie utilizată. Acest tip de operatori se aplică unuia sau mai multor biți dintr-o expresie

Operatorul unar “ $\sim$ ” returnează operandul asupra căruia este folosit cu toți biții asupra cărora s-a efectuat negarea.

Exemplu :

$\sim 0$ are valoarea 1	$0_{10} = 00000000_2$
prin negare bit cu bit se transformă în	$\sim 0_{10} = 11111111_2$

Operatorul binar “ $\&$ ” se utilizează pentru a transforma în 0 anumiți biți, sau pentru a testa o anumită configurație de biți.



Exemplu:

expresie	valoare
$1 \& 2$	0
$1 \& 3$	1

$$1_{10} = 00000001_2 \quad 1_{10} = 00000001_2$$

$$2_{10} = 00000010_2 \quad 3_{10} = 00000011_2$$


---

$$0_{10} = 00000000_2 \quad 1_{10} = 00000001_2$$

Operatorul binar “ $|$ ” se utilizează pentru a transforma în 1 anumiți biți.



Exemplu:

expresie	valoare
$1   2$	3
$1   3$	3

$$1_{10} = 00000001_2 \quad 1_{10} = 00000001_2$$

$$2_{10} = 00000010_2 \quad 3_{10} = 00000011_2$$


---

$$3_{10} = 00000011_2 \quad 3_{10} = 00000011_2$$

Operatorul binar “ $\wedge$ ” are următoarea tabelă de valori:



$\wedge$	0	1
0	0	0
1	1	0



Exemplu:

expresie	valoare
$1 \wedge 2$	3
$1 \wedge 3$	2

$$1_{10} = 00000001_2$$

$$1_{10} = 00000001_2$$

$$2_{10} = 00000010_2$$

$$3_{10} = 00000011_2$$

$$\hline$$

$$3_{10} = 00000011_2$$

$$\hline$$

$$2_{10} = 00000010_2$$

Operatorul “ $<<$ ” se folosește pentru a deplasa spre stânga biții unui număr întreg cu n biți specificați după operator; biții rămași în dreapta se completează cu 0.

Exemplu:  $2 << 2$

$$2_{10} = 00000010_2$$

$$\hline$$

$$8_{10} = 00001000_2$$

Deci, în general,  $a << n$  înseamnă, ca valoare,  $a * 2^n$ .

Operatorul “ $>>$ ” realizează deplasarea spre dreapta a biților unui număr întreg cu n poziții.

Deci, în general,  $a >> n$  înseamnă, ca valoare,  $a / 2^n$ , fiind de fapt împărțirea cu  $2^n$  a numărului.

## 6. Operatorul de atribuire “=”

Are cea mai mică prioritate după operatorul virgulă și apare sub forma:



var = expresie

Dacă avem var = var operator expresie, atunci în C/C++, se poate scrie prescurtat:

var operator = expresie.

unde operator poate să fie un operator aritmetic sau un operator logic pe biți.

Exemplu: int i, j;

i = i + 2;      se poate scrie i += 2;

j = j | 1;      se poate scrie j |= 1;

## 7. Operatori de incrementare ( ++ ) și decrementare ( -- )

	OPERATOR	FUNCTIE
	++i	incrementare prefixată
	-i	decrementare prefixată
	i++	incrementare postfixată
	i--	decrementare postfixată

De fapt:

$++i \Leftrightarrow i = i + 1$  - incrementarea se face înainte de evaluarea expresiei în care apare ++.

$i++ \Leftrightarrow i = i + 1$  - incrementarea se face după evaluarea expresiei în care apare ++.

$-i \Leftrightarrow i = i - 1$  - decrementarea se face înainte de evaluarea expresiei în care apare --.

$i- \Leftrightarrow i = i - 1$  - decrementarea se face după evaluarea expresiei în care apare --.

Observație: Acești operatori nu se aplică unor expresii.

Pentru o înțelegere mai eficientă diferența între cele două forme, prezintă mai jos o secvență de program care utilizează ambii operatori:



```
int valoare = 1;  
cout<<"Utilizarea formei postfixate "<<valoare++<<endl;  
cout<<"Valoarea dupa incrementare "<<valoare<<endl;  
valoare=1;  
cout<<"Utilizarea formei prefixate "<<++valoare<<endl;  
cout<<"Valoarea dupa incrementare "<<valoare<<endl;
```

Rezultatele execuției sunt următoarele:

*Utilizarea formei postfixate 1*  
*Valoarea dupa incrementare 2*  
*Utilizarea formei prefixate 2*  
*Valoarea dupa incrementare 2*

## 8. Operatorul de conversie explicită (cast)

Se folosește pentru forțarea tipului unui operand. Are forma:



**(tip) operand**

unde **operand** poate fi și o expresie, iar **tip** este un operator unar.



Exemplu: int i=5, j=2;  
float a;  
a = i / j; înseamnă că a primește valoarea întreagă 2  
a = (float) (i / j); înseamnă că a primește ca valoare 2  
a = (float) i / j; înseamnă că a primește ca valoare 2.5

## 9. Operatorul de lungime (sizeof)

Este un operator cu o prioritate mare, folosit în determinarea lungimii în octeți a unui operand care poate fi o expresie sau un tip de date.



**sizeof (expresie)**



**sizeof (tip)**

Pentru a putea înțelege eficiența folosirii acestui operator, prezentăm un scurt program:



```
cout<<"Variabila de tip int foloseste "<<sizeof(int)<<" octeti
"<<endl;
cout<<"Variabila de tip float foloseste "<<sizeof(float)<<
" octeti"<<endl;
cout<<"Variabila de tip double foloseste "<< sizeof(double)<<
" octeti"<<endl;
cout<<"Variabila de tip unsigned foloseste
"<<sizeof(unsigned)<<" octeti"<<endl;
```

În funcție de compilatorul C/C++ folosit se pot obține următoarele rezultate:

*Variabila de tip int folosește 2 octeți*  
*Variabila de tip float folosește 4 octeți*  
*Variabila de tip double folosește 8 octeți*  
*Variabila de tip unsigned folosește 2 octeți*  
*Variabila de tip long folosește 4 octeți*

## 10. Operatorii condiționali "?" ":"

Se folosesc numai împreună și funcționează ca o instrucțiune *if*.

Forma este următoarea:



**expresie<sub>1</sub> ? expresie<sub>2</sub> : expresie<sub>3</sub>**

sau



**var = expresie<sub>1</sub> ? expresie<sub>2</sub> : expresie<sub>3</sub>**

unde expresie<sub>1</sub> este o expresie logică care poate fi adevarată sau falsă; când este adevarată se evaluează expresie<sub>2</sub>, iar când este falsă se evaluează expresie<sub>3</sub>.



Exemplu:

```
modul = x < 0 ? -x : x;  
max = a < b ? b : a;  
min = a < b ? a : b;
```

## 11. Operatorul virgula „,”

Are prioritatea cea mai scazută din toți operatorii. Are următoarea formă:



**expresie<sub>1</sub>, expresie<sub>2</sub>, expresie<sub>3</sub>, ... expresie<sub>n</sub>**

sau



**expresie = expresie<sub>1</sub>, expresie<sub>2</sub>, expresie<sub>3</sub>, ... expresie<sub>n</sub>**

Această expresie se evaluează de la stânga la dreapta, evaluându-se pe rând fiecare expresie până se ajunge la ultima.

## 12. Tabela de priorități a operatorilor

În limbajul C/C++, pentru a evalua expresiile în care apar operatori, s-au stabilit anumite *reguli de prioritate (precedență) a operatorilor*.



NIVEL DE PRIORITATE	OPERATOR	ASOCIAȚIVITATE
1	0 [] . ->	stânga -> dreapta
2	- (unar) + (unar) & (unar) ! ~ *(unar) (cast) (sizeof) ++ --	dreapta -> stânga
3	* (binar) / %	stânga -> dreapta
4	+ (binar) - (binar)	stânga -> dreapta
5	<< >>	stânga -> dreapta
6	< <= > >=	dreapta -> stânga
7	== !=	stânga -> dreapta
8	&	stânga -> dreapta
9	^	stânga -> dreapta
10		stânga -> dreapta
11	&&	stânga -> dreapta
12		stânga -> dreapta
13	? :	stânga -> dreapta
14	=	dreapta -> stânga
15	,	stânga -> dreapta

## 1.3. Instrucțiunile limbajului C++

Limbajul C/C++ are câteva instrucțiuni cu ajutorul cărora se pot construi programe. Acestea sunt:

Instrucțiunea vidă:



Instrucțiunea compusă:



este delimitată de { și se termină cu }.

Uneori programele trebuie să efectueze una sau mai multe instrucțiuni atunci când o condiție este îndeplinită (într-o instrucțiune if) și altele când condiția este falsă. Limbajul C consideră instrucțiunile ca fiind *instrucțiuni simple* și *instrucțiuni compuse*.

O *instrucțiune simplă* este o singură instrucțiune, cum ar fi acea de atribuire sau de apel al unei funcții standard (de exemplu funcția `printf`).

O *instrucțiune compusă* este alcătuită din două sau mai multe instrucțiuni incluse între accolade.

Instrucțiunea expresie:



**expresie;**

Are 3 forme:

a) instrucțiune de atribuire:



**variabila = expresie;**

sau

**variabila operator = expresie;**

b) instrucțiunea de apel de funcție:



**nume\_functie(pa<sub>1</sub>, pa<sub>2</sub>, . . . , pa<sub>n</sub>);**

unde p<sub>a1</sub>, p<sub>a2</sub>, . . . , p<sub>an</sub> sunt parametrii actuali ai funcției (adică valorile cu care se va lucra în funcția respectivă la apelul funcției).

c) instrucțiunea de incrementare / decrementare:



**variabila ++;**

**++ variabila;**

**variabila --;**

**-- variabila;**

## Instrucțiunea if (instrucțiune de decizie sau condițională)

Are două forme:

a)



**if(expresie) instructiune;**

Observatie: În limbajul C/C++, spre deosebire de limbajul PASCAL, nu există cuvântul cheie **THEN**.

b)



**if(expresie)  
    instructiune1;  
else  
    instructiune2;**

**Observație:** Sunt permise folosirea instrucțiunilor **if** imbricate – adică **if** – uri cuprinse unul în corpul celuilalt. Nivelul de imbricare este nelimitat.

Din declarația unei instrucțiuni if se observă că, mai întâi, se evaluează expresia dată, iar dacă este adevarată atunci se execută instrucțiunea1, altfel instrucțiunea2. Fiecare din cele două instrucțiuni poate să fie simplă sau compusă.

## Instrucțiunea repetitivă while



**while(expresie) instructiune;**

unde instrucțiune poate fi instrucțiunea vidă, instrucțiunea simplă sau instrucțiunea compusă.

Funcționarea unei astfel de instrucțiuni se bazează pe evaluarea condiției date și executarea repetată a instrucțiunii *cât timp* condiția este îndeplinită.

### Exemplu:

Prezentăm în continuare un program în limbajul C/C++, care calculează suma primelor n numere naturale, cu  $n \leq 10$ :



```
int main(void)
{
    int i,n,s=0;
    cout<<"Dati numarul n = ";
    cin>>n;
    i=1;
    while(i<=n)
    {
        s+=i;
        i++;
    }
    cout<<"Suma primelor "<<n<<" numere naturale este
"<<s<<endl;
}
```

La execuția acestui program se poate obține următorul rezultat:

*Dati numarul n = 5*

*Suma primelor 5 numere naturale este 15*

Folosind facilitățile limbajului C/C++, se poate scrie aceeași secvență de program într-o formă prescurtată și chiar mai eficientă:



```
int main(void)
{
    int i=1,n,s=0;
    cout<<"Dati numarul n = "; cin>>n;
    while(i<=n) s+=i++;
    cout<<"Suma primelor "<<n<<" numere naturale este
"<<s<<endl;
}
```

La executia acestui program se poate obtine urmatorul rezultat:

*Dati numarul n = 5*

*Suma primelor 5 numere naturale este 15*

**Observație:** Pentru ca un ciclu repetitiv să se execute încontinuu (la infinit), se poate utiliza o buclă infinită prin introducerea ca expresie a unei valori diferite de 0.

**while (1)** este o buclă infinită care se va executa până când de la tastatură se va întrerupe executia prin apăsarea tastelor **CTRL+C**.

### Instrucțiunea repetitivă **for**

Este una dintre cele mai “puternice” instrucțiuni ale limbajului C/C++, datorită formei sale.



**for(expresie<sub>1</sub> ; expresie<sub>2</sub> ; expresie<sub>3</sub>) instructiune;**

Astfel:

**expresie<sub>1</sub>** – reprezintă secvența de inițializarea a ciclului

**expresie<sub>2</sub>** – reprezintă condiția de terminare a ciclului

**expresie<sub>3</sub>** – reprezintă secvența de reinicializare a ciclului

**instructiune** - corpul ciclului

Se stie că instrucțiunea **for** este de fapt o variantă particulară a instrucțiunii **while**, drept pentru care se poate scrie echivalent astfel:



**expresie<sub>1</sub>;  
while(expresie<sub>2</sub>)  
{  
    instructiune;  
    expresie<sub>3</sub>;  
}**

Invers, dacă avem:

**while (expresie) instructiune; ⇔ for( ; expresie ; ) instructiune;**

Funcționarea instrucțiunii **for** are loc astfel: Se pornește ciclul repetitiv prin inițializarea sa, adică prin execuția **expresia<sub>1</sub>**, iar apoi se evaluează

**expresia<sub>2</sub>** și dacă este adevărată se execută corpul ciclului, adică **instructiune**, după aceea se execută **expresia<sub>3</sub>**, și se reia evaluarea **expresiei<sub>2</sub>**, și a.m.d..

**Observație:** **expresia<sub>1</sub>**, **expresia<sub>2</sub>**, **expresia<sub>3</sub>** pot să lipsească, dar este obligatorie prezența semnelor: “;”.

**for(;;)**  $\Leftrightarrow$  **while(1)** – buclă infinită.

**Exemplu:** Același program de adunare a primelor n numere naturale, în varianta cu instrucțiunea **for**, va avea o dimensiune mai mică:



```
int main(void)
{
    int i,n,s=0;
    cout<<"Dati numarul n = ";
    cin>>n;
    for(i=1;i<=n;i++) s+=i;
    cout<<"Suma primelor "<<n<<" numere naturale este
"<<s<<endl;
}
```

La execuția acestui program se poate obține următorul rezultat:

*Dați numărul n = 5*

*Suma primelor 5 numere naturale este 15*

Sau mai eficient și mai scurt:



```
int main(void)
{
    int i,n,s;
    cout<<"Dati numarul n = ";
    cin>>n;
    for(i=1, s=0; i<=n; s+=i, i++);
    cout<<"Suma primelor "<<n<<" numere naturale este
"<<s<<endl;
}
```

La execuția acestui program se poate obține următorul rezultat:

*Dați numărul n = 5*

*Suma primelor 5 numere naturale este 15*

## Instrucțiunea repetitivă do while



```
do  
    instructiune;  
while(expresie);
```

unde instrucțiune poate fi instrucțiunea vidă, instrucțiunea simplă sau instrucțiunea compusă.

Funcționarea unei astfel de instrucțiuni se bazează pe executarea repetată a instrucțiunii *cât timp* condiția este îndeplinită.

Echivalența cu instrucțiunea *while*:



```
instructiune;  
while(expresie)  
    instructiune;
```

Echivalența cu instrucțiunea *for*:



```
for( instructiune; expresie ; ) instructiune;
```

Prezentăm în continuare o problemă:

Să se scrie un program care tipărește numerele naturale de la 0 la 9 și suma lor pe parcurs.



```
int main(void)  
{  
    int numar=0, total=0;  
    do{  
        total+=numar;  
        cout<<"numar" = "<<numar++<<" total = "  
    }<<total<<endl;  
    }while(numar<10);  
}
```

La execuția acestui program se obține următorul rezultat:

*numar=0 total=0*

*numar=1 total=1*

*numar=2 total=3*

```
numar=3 total=6
numar=4 total=10
numar=5 total=15
numar=6 total=21
numar=7 total=28
numar=8 total=36
numar=9 total=45
```

### Instrucțiunea break



Instrucțiunea încrerupe execuția instrucțiunilor **while**, **do while**, **for** și **switch**, determinând astfel ieșirea forțată dintr-un ciclu repetitiv sau din switch.

#### Exemplu:

```
for(;;)
{
    ...
    break;
}
```

Exemplu: Prezentăm în continuare un program, care folosind instrucțiunea **break**, numără numerele întregi aflate între 1 și 100 și apoi de la 100 la 1. De fiecare dată când număr ajunge la valoarea 50, instucțiunea **break** face ca execuția ciclului să se opreasă:



```
int main(void){
    int numar;
    for(numar = 1; numar<=100; numar++)
    {
        if(numar == 50) break;
        cout<<“ ”<<numar;
    }
    cout<<endl<<“Cel de-al doilea ciclu repetitiv”;
    for(numar = 100; numar>=1; numar--)
    {
        if(numar == 50) break;
        cout<<“ ”<<numar;
    }
}
```

## Instrucțiunea switch



```
switch (expresie)
{
    case c1 : sir_instructiuni_1;
                break;
    case c2 : sir_instructiuni_2;
                break;
    .....
    case cn : sir_instructiuni_n;
                break;
    default : sir_instructiuni;
}
```

Instrucțiunea **switch** funcționează astfel : Se evaluează expresia și în funcție de rezultat se compară cu  $c_1, c_2, \dots, c_n$  și când expresia este egală cu  $c_1$  atunci se execută șirul de instrucțiuni corespunzător, iar cu instrucțiunea **break** se sare la sfârșitul instrucțiunii **switch**. Același lucru se întâmplă și dacă expresia este egală cu  $c_2$ , sau cu  $c_3$ , sau cu  $c_n$ .

Instrucțiunea **switch** este o instrucțiune de tip decizie multiplă astfel încât se poate scrie echivalent folosind instrucțiunea de decizie simplă **if**:



```
if (expresie==c1)
    sir_instructiuni_1;
else if (expresie==c2)
    sir_instructiuni_2;
.....
else if (expresie==cn)
    sir_instructiuni_n;
else sir_instructiuni;
```

**Exemplu:** Să se scrie un program care să afișeze și să numere toate caracterele introduse de la tastatură până se apasă tasta “i”:



```
#include<iostream.h>
void main(void)
{
    int n=0;
    char a;
    switch(a==getche())
    {
        case (a!='i'):
        {
            printf("caracterul este %c\n",a);
            n++;
            printf("numarul de caractere n = %d",n);
        } break;
        default : printf("a-ti apasat i \n");
    }
}
```

**Exemplu:** Prezentăm în continuare un program care numără vocalele și consoanele din alfabet. De observat că unele din instrucțiunile *case* se execută în cascadă, pentru calculul vocalelor, iar pentru consoane se folosește cazul *default*.



```
int main(void)
{
    char litera;
    int nr_vocale = 0, nr_consoane = 0;
    for (litera = 'A'; litera <= 'Z'; litera++)
        switch (litera) {
            case 'A':
            case 'E':
            case 'I':
            case 'O':
            case 'U': nr_vocale++;
            break;
            default: nr_consoane++;
        }
    cout<<"Numarul de vocale este "<<nr_vocale<<endl;
    cout<<"Numarul de consoane este "<<
    nr_consoane<<endl;
}
```

**Problemă rezolvată:** Să se determine selectiv suma  $S_k=1^k+2^k+\dots+n^k$ , unde  $k=1, 2, 3, 4$ , cu preluarea de la tastatură a lui  $n$  și  $k$ .



```
int main(void)
{
    int numar;
    printf("\n Numerele pare dintre 1 si 100 sunt: \n");
    for (numar = 1; numar <= 100; numar++)
    {
        if(numar % 2) continue;
        cout<<" "<<numar;
    }
    cout<<endl<<" Numerele impare dintre 1 si 100 sunt :
"<<endl;;
    numar=0;
    while(numar <= 100)
    {
        numar++;
        if( ! numar % 2) continue;
        cout<<" "<<numar;
    }
}
```

### Instrucțiunea continue

Se referă la instrucțiunile de ciclare: **for**, **while** și **do while**. La întâlnirea ei ciclurile **while** și **do while** se continuă cu reevaluarea condiției de ciclare iar în ciclul **for** se continuă cu secvența de reinicializare a ciclului și apoi cu reevaluarea ciclului.

**Exemplu:** Prezentăm în continuare, un program care folosind instrucțiunea *continue* într-un ciclu *for* și într-un ciclu *while*, afișează numerele pare și impare aflate între 1 și 100:



```
#include <iostream.h>
#include <conio.h>
int main(void)
{
    int n,k;
    char raspuns;
    do
    {
        cout<<endl<<" introduceti n - maxim: ";
        cin>>n;
        cout<<" Introduceti puterea ( 1,2,3,4 ) : ";
        cin>>k;
        switch(k)
        {
            case 1 : cout<<endl<<" S_i**1 = "<<n*(n+1)/2;break;
            case 2 : cout<<endl<<" S_i**2 =
"<<n*(n+1)*(2*n+1)/6; break;
            case 3 : cout<<endl<<" S_i**3 =
"<<n*n*(n+1)*(n+1)/4; break;
            case 4 : cout<<" S_i**4 = "<<n * (n+1) *
(6*n*n*n+9*n*n+n-1)/30); break;
            default : cout<<endl<<" putere in afara intervalului
!";
        }
        cout<<endl<<endl<<" Doriti sa continuati ? ( d / n )";
        raspuns=getchar();
    }while(raspuns=='d' || raspuns=='D');
    cout<<endl<<" iesire din program";
    getch();
}
```

### Instrucțiunea goto

Este instrucțiunea pentru salt necondiționat.



**goto eticheta;**

unde eticheta este un nume care prefixează o instrucțiune.

**Exemplu:** Prezentăm în continuare, un program care folosind instrucțiunea *goto*, afișează numerele întregi aflate între 1 și 100:

```
int main(void)
{
    int numar=1;
    eticheta: cout<< " ", numar++;
    if (numar <= 100) goto eticheta;
}
```

### Instrucțiunea return

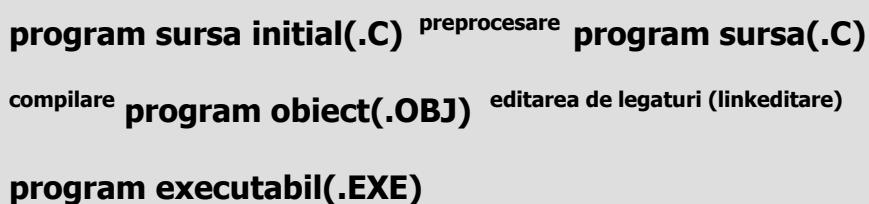


1. **return;**
2. **return expresie;**
3. **return (expresie);**

Se folosește în funcții atunci când se întoarce în funcția apelantă o valoare (formele 2 și 3) sau într-o funcție care nu întoarce nici o valoare (funcționează ca o procedură) – forma 1.

### Preprocesare

Schematic, parcursul unui program sursă scris în C, până la executarea lui este următorul:



În C++, după construirea unui program sursă, se pot evalua anumite valori dacă se utilizează preprocesorul. Activitatea preprocesorului se împarte în:

- a) includere fișiere
- b) definire constante simbolice

- c) definire macrouri
- d) compilare condiționată

a) **Includerea de fișiere** se face pentru a putea utiliza funcțiile predefinite ale limbajului C++ și care se află în fișiere standard numite **header-e** (au extensia .h), sau pentru a putea utiliza funcții proprii aflate în fișiere utilizator.

Astfel, pentru a putea utiliza funcțiile standard de intrare / ieșire (**scanf** / **printf**), trebuie scris la începutul unui program C++:

**#include <stdio.h>** - fișier standard

iar pentru a utiliza fișiere utilizator:

**#include "nume fisier"** – fișier utilizator

Observatie: Includerea fișierelor se execută numai pe timpul compilării.

### b) Definire constante simbolice

Pentru a mări portabilitatea programelor C++, se pot folosi *constante*. O **constantă** este un nume pe care compilatorul C++ îl asociază unei valori care nu se modifică. Pentru aceasta se utilizează directiva **#define**.



**#define nume\_constanta text**

Asociază numelui **nume\_constanta** textul denumit **text** care se poate prelungi pe mai linii. Aceasta substituie a numelui este valabilă în tot fișierul până la întâlnirea unei directive de compilare **#undef nume**.

Exemplu:

```
#define NRLINII 30
#define NRCOLOANE 20
#define DIMENSIUNE NRLINII*NRCOLOANE
int t[NRLINII][NRCOLOANE];
double a[DIMENSIUNE];
```

Ce va întâlni compilatorul după procesor?

```
int t[30][20];
double a[30*20];
```

Observație: nu se vor face calculele pentru că este vorba doar de texte.

Dacă se vor utiliza macroinstructiuni sau constante simbolice în programele C, atunci acestea trebuie să aibă nume sugestive și scrise cu litere mari pentru a putea ajuta programatorii care citesc codul sursă să facă diferența ușor între constante și variabile.

Exemplu: Putem defini următoarele constante:

```
#define TRUE 1  
#define FALSE 0  
#define PI 3.1415  
#define PROGRAMATOR "Runceanu Adrian"
```

### c) Definirea de macroinstructiuni

Un macrou (o macroinstructiune) este o definiție în care funcționează reguli de substituție de text.



```
#define nume(param_form1, param_form2, . . . ,  
param_formn) text
```

Apelul macroului se face astfel:



```
nume(param_actual1, param_actual2, . . . ,param_actualn)
```

Unde **nume** reprezintă numele macroului, **param\_form<sub>1</sub>**, **param\_form<sub>2</sub>**, . . . , **param\_form<sub>n</sub>** parametrii formali ai macroului, iar **text** este textul în care se vor face înlocuirile, iar **param\_actual<sub>1</sub>**, **param\_actual<sub>2</sub>**, . . . , **param\_actual<sub>n</sub>** sunt parametrii cu care se apeleză macroul în funcția principală.

Funcționarea macroului este următoarea: în textul dat se înlocuiesc parametrii formali cu parametrii actuali iar apoi textul obținut va substitui apelul macroului.



Exemplu:

```
#define MAX(a,b) ((a) < (b) ? (b) : (a))  
#define MIN(a,b) ((a) > (b) ? (b) : (a))  
#define SGN(x) ((x) > 0 ? 1 : ((x) == 0 ? 0 : (-1)))
```

Am definit câteva macrouri care pot fi utile în multe programe C++, și anume MAX și MIN determină *maximul*, respectiv *minimul* a două numere de orice tip, iar SGN determină *signatura* unui număr dat.

Următorul program C++ va folosi aceste două macrouri:



```
int main(void)
{
    printf("Maximul valorilor 10.0 si 25.0 este %f\n",MAX(10.0,25.0));
    printf("Minimul valorilor 3.4 si 3.1 este %f\n",MIN(3.4,3.1));
}
```

Atunci când va executa acest program, preprocesorul va avea ca rezultat următorul cod:

```
printf("Maximul valorilor 10.0 si 25.0 este %f\n", ((10.0) < (25.0)) ? (10.0) : (25.0));
printf("Minimul valorilor 3.4 si 3.1 este %f\n", ((3.4) > (3.1)) ? (3.4) : (3.1));
```

## 1.4. Probleme rezolvate

### 1.4.1. Enunțuri

1. Să se determine cel mai mare divizor comun (*c.m.m.d.c.*) și cel mai mic multiplu comun (*c.m.m.m.c.*) a două numere întregi citite de tastatură. C.m.m.d.c. se va calcula folosind cele două variante:

- algoritmul lui *Euclid*
- folosind relația de mai jos:

$$\text{cmmdc}(a,b) = \begin{cases} \text{cmmdc}(a-b,b), & \text{dacă } a > b \\ \text{cmmdc}(a,b-a), & \text{dacă } a < b \\ a, & \text{dacă } a=b \end{cases}$$

2. Să se determine toți divizorii unui număr întreg citit de la tastatură.

3. Să se verifice dacă un număr întreg este număr *prim* sau nu. Spunem că un număr  $n$  este număr prim dacă are ca divizori naturali numai valorile 1 și  $n$ .

4. Să se verifice dacă un număr este număr *perfect* sau nu. Spunem că un număr este număr perfect dacă este egal cu suma divizorilor lui, mai puțin el însuși. (Exemplu: numărul 6 este perfect, deoarece este egal cu suma divizorilor săi 1,2,3).

5. Se citesc  $n$  numere întregi. Să se determine minimul și maximul lor.

6. Să se calculeze suma cifrelor unui număr întreg.

7. Să se verifice dacă un număr este *palindrom* sau nu. Spunem că un număr este palindrom dacă este egal cu răsturnatul său (adică numărul format din cifrele de la dreapta la stânga ale numărului inițial – exemplu : n = 25652).

8. Să se scrie un program care să folosească variabile de tip întreg, cărora să li se atribuie valori prin program, să se adune aceste variabile, să se calculeze media lor aritmetică, rezultatul să se depună într-o variabilă de tip întreg și să se afișeze.

9. Să se modifice programul anterior, în sensul că cele două valori se vor citi de la tastatură.

10. Să se calculeze și să se afișeze valoarea distanței între două puncte, dându-se coordonatele acestora: A(x1, y1) și B(x2, y2).

11. De la tastatură să se citească valoarea unui unghi sub forma de grade. Să se calculeze și să se afișeze unghiul în radiani.

12. Se citesc de la tastatură trei valori a, b, c. Să se afișeze, în ordine crescătoare, valorile variabilelor, fără modificarea conținutului acestora.

În funcție de valorile citite se va afișa una din situațiile:

a b c  
a c b  
c a b  
c b a  
b a c  
b c a

13. Să se modifice programul anterior, astfel încât valorile variabilelor să poată fi schimbate.

14. Să se scrie un program care să rezolve ecuația de grad I:  $ax+b=0$ , valorile lui  $a$  și  $b$  se citesc de la tastatură și sunt valori reale. Se vor lua în discuție toate cazurile.

15. Să se scrie un program care să rezolve ecuația de grad al II-lea:  $ax^2+bx+c=0$ , unde  $a, b, c \in \mathbb{R}$ . Se vor lua în discuție toate cazurile.

### 1.4.2. Soluții propuse

#### Problema 1:

Să se determine cel mai mare divizor comun (*c.m.m.d.c.*) și cel mai mic multiplu comun (*c.m.m.m.c.*) a două numere întregi citite de tastatură. C.m.m.d.c. se va calcula folosind cele două variante:

- algoritmul lui *Euclid*
- folosind relația de mai jos:

$$\text{cmmdc}(a,b) = \begin{cases} \text{cmmdc}(a-b,b), & \text{dacă } a > b \\ \text{cmmdc}(a,b-a), & \text{dacă } a < b \\ a, & \text{dacă } a=b \end{cases}$$



```
#include <iostream.h>
int main(void)
{
    int a, b, x, y, r, cmmdc1, cmmdc2, cmmmc;
    cout<<"Dati primul numar "; cin>>a;
    cout<<"Dati al doilea numar "; cin>>b;
    // calculam c.m.m.d.c. folosind algoritmul lui Euclid
    x = a; y = b;
    r = a % b;
    while( r != 0)
    {
        a = b;
        b = r;
        r = a % b;
    }
    cmmdc1 = b;
    cout<<"Cmmdc este "<<cmmdc1<<endl;
```

```

// calculam c.m.m.d.c. folosind relatia data
while( a != b )
    if( a > b )
        a=a-b;
    else
        if( a < b )
            b=b-a;
    cmmdc2 = a;
    cout<<"Cmmdc este "<<cmmdc2<<endl;
    cmmmc = ( x * y ) / cmmdc1;
    cout<<"Cmmmc este "<<cmmmc<<endl;
}

```

### Problema 2:

Să se determine toți divizorii unui număr întreg citit de la tastatură.



```

#include <iostream.h>
int main(void)
{
    int n, i;
    cout<<"Dati numarul "; cin>>n;
    cout<<"Divizorii numarului "<<n<<" sunt: ";
    for(i=1; i<=n; i++)
        if( n % i == 0)
            cout<<" "<<i;
}

```

### Problema 3:

Să se verifice dacă un număr întreg este număr *prim* sau nu. Spunem că un număr  $n$  este număr prim dacă are ca divizori naturali numai valorile 1 și  $n$ .



```

#include <iostream.h>
int main(void)
{
    int n,i,prim;
    cout<<"Dati numarul "; cin>>n;
    prim=1;
    for(i=2;i<=n/2;i++)
        if(n%i==0) prim=0;
    if(prim==1)
        cout<<"Numarul "<<n<<" este numar PRIM";
    else
        cout<<"Numarul "<<n<<" NU este numar PRIM";
}

```

#### Problema 4:

Să se verifice dacă un număr este număr *perfect* sau nu. Spunem că un număr este număr perfect dacă este egal cu suma divizorilor lui, mai puțin el însuși. (Exemplu: numărul 6 este perfect, deoarece este egal cu suma divizorilor săi 1,2,3).



```
#include <iostream.h>
int main(void)
{
    int n, i, s = 0;
    cout<<"Dati numarul  "; cin>>n;
    for(i=1; i<=n/2; i++)
        if( n % i == 0 ) s+=i;
    if( n == s )
        cout<<"Numarul    "<<n<<"    este    numar
PERFECT";
    else
        cout<<"Numarul    "<<n<<"    NU    este    numar
PERFECT";
}
```

#### Problema 5:

Se citesc n numere întregi. Să se determine minimul și maximul lor.



```
#include <iostream.h>
int main(void)
{
    int n, i, max, min, x;
    min = 32367;
    max = -32368;
    cout<<"Dati numarul  "; cin>>n;
    for(i=1; i<=n; i++)
    {
        cout<<"dati numarul "<<i<<" ";
        cin>>x;
        if( max < x ) max=x;
        if( min > x ) min=x;
    }
    cout<<"Maximul este "<<max<<endl;
    cout<<"Minimul este "<<min;
}
```

### Problema 6:

Să se calculeze suma cifrelor unui număr întreg.



```
#include <iostream.h>
int main(void)
{
    int n, i, suma=0, r;
    cout<<"Dati numarul  ";  cin>>n;
    while( n != 0 )
    {
        r = n % 10;
        suma+= r;
        n = n / 10;
    }
    cout<<"Suma este "<<suma<<endl;
}
```

### Problema 7:

Să se verifice dacă un număr este *palindrom* sau nu. Spunem că un număr este palindrom dacă este egal cu răsturnatul său (adică numărul format din cifrele de la dreapta la stânga ale numărului inițial – exemplu: n = 25652).



```
#include <iostream.h>
int main(void)
{
    int n, r, m = 0, x;
    cout<<"Dati numarul  ";  cin>>n;
    x = n;
    while( n != 0 )
    {
        r = n % 10;
        m+= r;
        n = n / 10;
    }
    if( x == m )
        cout<<"Numarul "<<x<<" este PALINDROM"<<endl;
    else
        cout<<"Numarul "<<x<<" NU este PALINDROM"<<endl;
}
```

### Problema 8:

Să se scrie un program care să folosească variabile de tip întreg, cărora să li se atribuie valori prin program, să se adune aceste variabile, să se calculeze media lor aritmetică, rezultatul să se depună într-o variabilă de tip întreg și să se afișeze.



```
#include<iostream.h>
int main(void)
{
    int a, b, suma, media;
    a = 5; b = 7;
    suma = a + b;
    media = suma / 2;
    cout<<"media este ="<<media;
}
```

### Problema 9:

Să se modifice programul anterior, în sensul că cele două valori se vor citi de la tastură.



```
#include<iostream.h>
int main(void)
{
    int a, b, suma, media;
    cout<<"dati primul numar a=";
    cin>>a;
    cout<<"dati al doilea numar b=";
    cin>>b;
    suma = a + b;
    media = suma / 2;
    cout<<"media este ="<<media;
}
```

### Problema 10:

Să se calculeze și să se afișeze valoarea distanței între două puncte, dându-se coordonatele acestora: A(x<sub>1</sub>, y<sub>1</sub>) și B(x<sub>2</sub>, y<sub>2</sub>).



```
#include<iostream.h>
#include<math.h>
int main(void)
{
    int x1, y1, x2, y2;
    float d;
    cout<<"dati x1= "; cin>>x1;
    cout<<"dati y1= "; cin>>y1;
    cout<<"dati x2= "; cin>>x2;
    cout<<"dati y2= "; cin>>y2;
    d = sqrt( (x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2) );
    cout<<"Distanta intre "<<x1<<","<<y1<<" si
"<<x2<<","<<y2<<" este = "<<d;
}
```

### Problema 11:

De la tastatură să se citească valoarea unui unghi sub forma de grade.  
Să se calculeze și să se afișeze unghiul în radiani.



```
#include<iostream.h>
int main(void)
{
    int u;
    float r;
    cout<<"dati unghiul u = "; cin>>u;
    r = (float)u / 180;
    cout<<"unghiul in radiani este = pi*"<<r;
}
```

### Problema 12:

Se citesc de la tastatură trei valori a, b, c. Să se afișeze, în ordine crescătoare, valorile variabilelor, fără modificarea conținutului acestora.

În funcție de valorile citite se va afișa una din situațiile:

a b c

a c b

c a b

c b a

b a c

b c a



```
#include<iostream.h>
int main(void)
{
    int a, b, c;
    cout<<"Dati primul numar = "; cin>>a;
    cout<<"Dati al doilea numar = "; cin>>b;
    cout<<"Dati al treilea numar = "; cin>>c;
    if ( a < b )
        if( b < c ) cout<<a<<", "<<b<<", "<<c;
        else if( a < c ) cout<<a<<", "<<c<<", "<<b;
        else cout<<c<<", "<<a<<", "<<b;
        else if( a < c ) cout<<b<<", "<<a<<", "<<c;
        else if(b < c )
            cout<<b<<", "<<c<<", "<<a;
            else
                cout<<c<<", "<<b<<", "<<a;
}
```

### Problema 13:

Să se modifice programul anterior, astfel încât valorile variabilelor să poată fi schimbată.



```
#include<iostream.h>
int main(void)
{
    int a, b, c, temp;
    cout<<"Dati primul numar = "; cin>>a;
    cout<<"Dati al doilea numar = "; cin>>b;
    cout<<"Dati al treilea numar = "; cin>>c;
    if( a > b )
    {
        temp = a;      a = b;  b = temp;
    }
    if( b > c )
    {
        temp = b;      b = c;  c = temp;
    }
    if( a > b )
    {
        temp = a;      a = b;  b = temp;
    }
    cout<<"Numerele in ordine crescatoare sunt
    "<<a<<", "<<b<<", "<<c;
}
```

### Problema 14:

Să se scrie un program care să rezolve ecuația de grad I:  $ax+b=0$ , valorile lui  $a$  și  $b$  se citesc de la tastatură și sunt valori reale. Se vor lua în discuție toate cazurile.



```
#include<iostream.h>
int main(void)
{
    float a,b,x;
    cout<<"Dati valoarea lui a = ";
    cin>>a;
    cout<<"Dati valoarea lui b = ";
    cin>>b;
    if( a == 0 )
        if( b == 0 ) cout<<"infinitate de solutii";
        else cout<<"ecuatie imposibila";
    else
    {
        x = -b / a;
        cout<<"solutia este x= "<<x;
    }
}
```

### Problema 15:

Să se scrie un program care să rezolve ecuația de grad II:  $ax^2+bx+c=0$ , unde  $a, b, c \in \mathbb{R}$ . Se vor lua în discuție toate cazurile.



```
#include<iostream.h>
#include<math.h>
int main(void)
{
    int a,b,c;
    float x1,x2,delta,preal,pimag;
    cout<<"Dati valoarea lui a = ";
    cin>>a;
    cout<<"Dati valoarea lui b = ";
    cin>>b;
    cout<<"Dati valoarea lui c = ";
    cin>>c;
    if(a==0)
        cout<<"Ecuatie de gradul I"<<"\n";
```

```
else
{
    delta=b*b-4*a*c;
    if(delta >= 0)
    {
        cout<<"Radacini reale: ";
        x1=(-b+sqrt(delta))/(2*a);
        x2=(-b-sqrt(delta))/(2*a);
        cout<<"x1 = "<<x1<<" x2 = "<<x2;
    }
    else
    {
        cout<<"Radacini complexe :";
        delta=-delta;
        preal=-b/(2*a);
        pimag=sqrt(delta)/(2*a);
        cout<<"x1 = "<<preal<<" +
        i*"<<pimag<<"\n";
        cout<<"x2 = "<<preal<<" - i*"<<pimag;

    }
}
```

## 1.5. Probleme propuse spre rezolvare

1. Scrieți un program care citește de la tastatură două numere întregi și afișează “adevărat” dacă primul număr este un multiplu al celui de-al doilea și “fals” în caz contrar.
  
2. Scrieți un program care citește de la tastatură două numere întregi reprezentând o dată calendaristică (lună și zi) și afișează “adevărat” dacă ea coincide cu data Crăciunului și “fals” în caz contrar.
  
3. Scrieți un program care să testeze dacă un an este sau nu bisect. Dacă da, afișează “adevărat”, dacă nu, afișează “fals”. Precizare: anii bisecți sunt, în general, multipli de 4, dacă anii multipli de 100 care nu sunt și multipli de 400 nu sunt bisecți. De exemplu, anul 2000 este an bisect, în timp ce anii 1700, 1800, 1900 nu sunt ani bisecți.
  
4. Să se scrie un program care să citească de la tastatură trei valori întregi, pozitive, strict mai mici decât 1000 pe care apoi să le tipărească împreună cu media lor aritmetică sub forma:  
$$A = *** \quad B = *** \quad C = *** \quad MEDIA = ***.**$$
(S-a notat prin \* o poziție în care se va afișa un caracter ce face parte din reprezentarea valorii numerice respective)
  
5. Considerând R - raza unui cerc, LungCerc - lungimea cercului și Aria - suprafața acestuia (valori numerice de tip real), să se scrie secvența de instrucțiuni necesară pentru afișarea următoarelor informații:

Raza cercului : \*\*\*\*.\*\*\*

Lungimea cercului : \*\*\*\*.\*\*\*

Aria cercului : \*\*\*\*\*.\*\*\*\*

6. Scrieți un program care citește de la tastatură un număr și afișează sub formă de tabel numărul respectiv, numărul<sup>2</sup>, numărul<sup>3</sup>, numărul<sup>4</sup>.
7. O dată calendaristică introdusă de la tastatură sub forma a trei valori întregi (zi, lună, an) trebuie afișată pe ecran sub forma **zi/lună/an**, păstrând din valoarea anului doar ultimele două cifre. Scrieți fragmentul de program care implementează această funcție.
8. Două intervale de timp sunt exprimate în ore, minute și secunde. Să se calculeze suma lor exprimată în același mod.
9. Se citește un număr natural par. Se cere să se descompună în sumă de câte două numere prime (*conjectura lui GOLDBACH*).
10. Se citesc două numere naturale formate din cinci cifre distincte. Să se determine care este numărul cifrelor comune celor două numere (care nu concid obligatoriu și ca poziție).
11. Se consideră *șirul lui Fibonacci*, definit astfel:

$$fib(n) = \begin{cases} 1, & \text{daca } n=1 \\ 1, & \text{daca } n=2 \\ fib(n-1) + fib(n-2), & \text{daca } n > 2 \end{cases}$$

Pentru o valoare  $n$  dată să se afișeze toți termenii șirului lui Fibonacci mai mici sau egali cu termenul al  $n$ -lea.

*Exemplu :* Pentru  $n=7$ , se va afișa  $1, 1, 2, 3, 5, 8, 13$

12. Se dă un număr natural  $n > 0$ , scris în baza 10. Se cere să se afișeze reprezentările sale în bazele 2, 8 și 16.
13. Se consideră un număr natural  $n$ . Să se scrie un program C++ care să afișeze toate descompunerile numărului dat în sumă de numere consecutive.

*Exemplu:* Pentru  $n = 10 \Rightarrow 1 + 2 + 3 + 4$

$$n = 15 \Rightarrow 1 + 2 + 3 + 4 + 5 = 4 + 5 + 6 = 7 + 8$$

**14.** Scrieți un program care va citi o singură literă de la tastatură și va afișa cifra corespunzătoare de pe tastele unui telefon. Cifrele și literele de pe telefon sunt asociate după cum urmează:

2 = A ,B, C

3 = D, E, F

4 = G, H, I

5 = J, K , L

6 = M, N, O

7 = P , R, S

8 = T, U ,V

9 = W, X, Y

**15.** Să se verifice dacă un an calendaristic este bisect sau nu. Un an este bisect dacă :

- este divizibil cu 4 și nu este divizibil cu 100

sau

- este divizibil cu 400

*Exemplu:* Anii 2000, 1984 au fost bisecți, iar anii 1900, 1975 nu au fost bisecți.

**16.** Să se verifice dacă trei numere naturale date  $a$ ,  $b$  și  $c$  sunt pitagorice (adică dacă verifică una dintre condițiile  $a^2=b^2+c^2$ ,  $b^2=a^2+c^2$ ,  $c^2=a^2+b^2$ ).

*Exemplu:* numerele 3, 4, 5 sunt pitagorice.

**17.** Scrieți un program de convertire a unităților de măsură din sistemul englez în sistemul metric sau invers. Unitățile luate în calcul vor fi:

1. Inch (in)    2. Foot (ft)    3. Pound(lb)  4. Ounce(oz)  5. Gallon(gal)

respectiv

1. Centimetru(cm)    2. Metru    3. Kilogram (kg)    4. Gram (g)  5.

Litru(l)

Vor fi utilizați următorii factori de conversie:

1 in = 2,54 cm

1 lb = 0,45359 kg

1 gal = 3,78541 l

1 ft = 0,3048 m

1 oz = 28,3495 g

*Exemplu:* 13 in = 33.02 cm = 0.3302 m

**18.** Să se calculeze diferența a două unghiuri date în grade, minute și secunde.

*Exemplu:*  $30^{\circ} 50' 45'' - 10^{\circ} 20' 53'' = 20^{\circ} 29' 52''$

**19.** Se citesc trei numere întregi  $a, b, c$ . Să se verifice dacă aceste numere (pusă în orice ordine) sunt în progresie aritmetică și afișați rația progresiei în caz afirmativ.

*Exemplu:* {10, 3, 17} este o progresie aritmetică cu rație 7;  
{1, 2, 4} nu este o progresie aritmetică.

**20.** Să se scrie un program pentru transcrierea în cuvinte a unei valori întregi de patru cifre, introduse de la tastatură.

*Exemplu:*

1890: o mie opt sute nouăzeci

2004: două mii patru

3000: trei mii

1734: o mie sapte sute treizeci și patru

**21.** Să se determine ultima cifră a numărului  $a^b$ , pentru  $a$  și  $b$  numere naturale mai mici sau egale cu 32000, citite de la tastatură.

*Exemplu:* numărul  $124^{59}$  se termină cu cifra 4.

**22.** Să se afle toate numerele naturale mai mici decât 2000, care împărțite la 24, 30, 18 dau restul 7.

*Exemplu:*

$1447 : 24 = 60$  rest 7;

$1447 : 30 = 48$  rest 7;

$1447 : 18 = 80$  rest 7.

**23.** Să se scrie un program care determină cel mai mic număr care are exact  $k$  divizori.

*Exemplu:*

Pentru  $k=4$  se obține numărul 6.

**24.** Determinați cel mai mic număr  $\leq n$  care are numărul maxim de divizori proprii (divizorul propriu, este diferit de 1 și de el însuși).

*Exemplu:* Pentru  $n=20$  se obține numărul 12 (numerele 18 și 20 au același număr de divizori).

**25.** Pentru un număr  $n$  citit de la tastatura  $1 \leq n \leq 1000$ , să se scrie un program care să afișeze multimea divizorilor săi naturali (inclusive 1 și  $n$ ). De asemenea se vor mai afișa numerele:

$\tau(n)$  = numărul divizorilor lui  $n$

$\sigma(n)$  = suma divizorilor lui  $n$

*Exemplu:* Divizorii lui  $n = 20$  sunt 1, 2, 4, 5, 10, 20, suma lor este 42, iar numărul lor este 6.

**26.** Să se scrie un program care să calculeze câte perechi de numere naturale care nu depășesc un număr natural dat au cel mai mare divizor comun un număr dat  $d$ .

*Exemplu:* Pentru  $n = 20$  și  $d = 5$ , există 6 perechi (5,5), (5,10), (5,15), (5,20), (10,15), (15,20) care au cel mai mare divizor comun egal cu 5

**27.** Să se determine un număr natural de două cifre al cărui cub are șase cifre și se scrie numai cu cifrele 6, 7 și 8.

*Exemplu:* Numărul căutat este 92 ( $92^3 = 778688$ ).

**28.** Să se transforme un număr din baza  $b < 10$  în baza 10.

*Exemplu:*  $352_8 = 234_{10}$ .

**29.** Să se transforme un număr din baza  $p$  în baza  $q$ , unde  $p, q < 10$ .

*Exemplu:*  $352_8 = 1414_5$ .

**30.** Să se inverseze (oglindească) un număr (care nu se termină cu cifra 0), adică numărul care are cifrele numărului inițial dar în ordinea de la dreapta la stânga.

*Exemplu:* Pentru numarul 10758 se obtine numarul 85701.

**31.** Se citește un număr natural de la tastatură. Câte cifre conține?

*Exemplu:* Numărul 12602 conține 5 cifre.

**32.** Pentru un număr natural cu cel mult nouă cifre, să se determine numărul de cifre distințe din care se compune.

*Exemplu:* Numărul 13221 are 3 cifre distințe.

**33.** Scrieți un program care afișează numerele naturale mai mici sau egale cu o valoare  $n$  dată, cu proprietatea că suma cifrelor lor este un număr prim.

*Exemplu:* Pentru  $n=25$  se vor afișa valorile 2, 3, 5, 7, 11, 23.

**34.** Dintre numerele naturale mai mici sau egale cu o valoare  $n$  dată, să se afișeze acelea care sunt divizibile cu suma cifrelor lor.

*Exemplu:* Pentru  $n=25$  se vor afișa valorile 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 18, 20, 21, 24

**35.** Un număr se numește *palindrom* dacă citit invers este același număr. Să se verifice dacă un număr citit de la tastatură este palindrom sau nu.

*Exemplu:* Numărul 12321 este palindrom.

**36.** Se citește un număr natural  $n$  de la tastatură. Să se calculeze și afișeze restul și câtul împărțirii lui  $n$  la suma cifrelor lui  $n$ .

*Exemplu:* Pentru  $n=894$  se obține câtul 42 și restul 12.

**37.** Se numește număr "*bine ordonat*" *descrescător/crescător* un număr natural, cu proprietatea că cifrele sale citite de la stânga la dreapta sunt în ordine descrescătoare/crescătoare. Scrieți un algoritm care verifică dacă un număr natural  $x$  dat este "bine ordonat" descrescător sau crescător, afișând un mesaj corespunzător

*Exemplu:* Numărul 5310 este bine ordonat descrescător iar 146 este bine ordonat crescător.

**38.** Să se verifice dacă un număr  $n$  este palindrom în baza 16. Numărul este introdus în baza 10.

*Exemplu:* Numărul 111281 este palindrom în baza 16. ( $111281_{10}=1B2B1_{16}$ )

**39.** Se introduc de la tastatură numere întregi formate din minim două cifre, până la întâlnirea numărului 0. Să se afișeze pe ecran câte numere care au cifra unităților mai mică decât cifra zecilor există.

*Exemplu:* Dacă se introduce sirul 25, 653, 2965, 211, 154, 1256, 0 se va afișa valoarea 3.

**40.** Se spune că  $n$  este *deosebit* dacă există un număr natural  $m$  astfel încât  $n = m + S(m)$ , unde  $S(m)$  este suma cifrelor lui  $m$ . Să se scrie un program care să verifice dacă un număr  $A$  dat este deosebit.

*Exemplu:* 1235 este deosebit ( $1235=1225+10$ ).

**41.** Să se afișeze toate numerele mai mici decât 100000 egale cu suma factorialelor cifrelor componente.

*Exemplu:* Un astfel de număr este 145 ( $145=1!+4!+5!$ ),

**42.** Se citește un număr întreg care va fi stocat într-o variabilă de tip longint. Să se afișeze un mesaj corespunzător dacă numărul are aspect de "deal" sau de "vale".

*Exemplu:* Numărul 2465320 are aspect de deal (urcă până la cifra 6 apoi coboară). Numărul 52108 are aspect de vale (coboară la cifra 0 apoi urcă).

**43.** Să se calculeze produsul a două numere naturale prin adunări repetate.

*Exemplu:*  $3^9=27$  ( $3+3+3+3+3+3+3+3$  sau  $9+9+9$ ).

**44.** Fie  $n$  valori întregi. Să se calculeze suma algebraică a valorilor de rang impar (primul, al treilea, al cincilea, ...) și produsul tururor valorilor diferite de zero.

*Exemplu:* Dacă se introduc următoarele 10 numere 2, -5, 47, 0, 32, 12, -6, 14, 0, -3 se va obține suma 75 și produsul -45480960.

**45.** Fie  $n$  valori de tip întreg. Să se calculeze raportul dintre suma elementelor strict pozitive și produsul tuturor celor  $n$  valori.

*Exemplu:* Pentru  $n=5$  și numerele 9, -2, 7, 4, -3 se găsește raportul 0.01322751.

**46.** Să se determine toate numerele de forma  $a^2 + b^3$ , cu

$$1 \leq a \leq 5 \text{ și } 1 \leq b \leq 5.$$

*Exemplu:* Un număr de forma dată este 68 ( $68 = 2^2 + 4^3$ ).

**47.** Pentru mulțimile  $A=\{1, 2, \dots, m\}$  și  $B=\{1, 2, \dots, n\}$ . cu  $m$  și  $n$  citite de la tastatură, să se afișeze elementele mulțimii  $A \times B = \{(a, b) | a \in A, b \in B\}$

*Exemplu:* Pentru  $m=3$  și  $n=2$ ,  $A \times B = \{(1,1), (1,2), (2,1), (2,2), (3,1), (3,2)\}$

**48.** Să se determine toate numerele întregi de trei cifre  $\overline{abc}$  cu proprietatea că  $\overline{abc} = a^3 + b^3 + c^3$

*Exemplu:* Un astfel de număr este 371 ( $371 = 3^3 + 7^3 + 1^3$  ).

**49.** Fiind dat un număr întreg pozitiv  $n$ , scrieți un program care să calculeze numărul de cifre zecimale necesare pentru a scrie valoarea lui  $n$ . De exemplu, pentru numărul 27 sunt necesare 2 cifre zecimale iar pentru numărul 5, una singură.

**50.** De la tastatură se introduce o listă de numere întregi pozitive. Se cere să se afișeze valoarea maximă depistată în listă.

**51.** Cunoscând valoarea  $n$ , număr întreg pozitiv introdus de la tastatură, să se calculeze și să se afișeze suma:

$$S = \sum_{k=1}^n (-1)^k * k!$$

52. Citindu-se de la tastatură numărul natural  $n$ , să se calculeze și să se afișeze lista puterilor pozitive ale lui 2 a căror valoare este cel mult egală cu  $n$ .

53. De la tastatură se introduce o dată calendaristică sub forma a trei întregi (zi, lună, an). Se cere să se afișeze data sub forma **zi-lună-an**, în care luna să apară cu numele ei și nu ca număr întreg.

54. Scrieți un fragment de program cu ajutorul căruia să se determine dacă un caracter dat este literă, cifră, spațiu, semn de punctuație sau “alt caracter”.

55. Să se calculeze coeficienții binomiali  $C_n^1, C_n^2, \dots, C_n^p$ , în care  $n$  și  $p$  sunt valori întregi pozitive citite de la tastatură ( $p \leq n$ ), știind că există următoarea relație de recurență:

$$C_n^k = \frac{n-k+1}{k} * C_n^{k-1} \quad C_n^0 = 1.$$

56. Pentru  $n$  cunoscut, să se calculeze  $f_n$ , termenul de rangul  $n$  din sirul lui Fibonacci, știind că:

$$f_0 = 1 ; f_1 = 1 ; f_p = f_{p-1} + f_{p-2} \text{ pentru orice valoare } p \geq 2.$$

57. Dându-se numărul întreg  $n$ , să se calculeze numărul întreg  $x$  format din  $n$  cifre citite pe rând de la tastatură într-o aceeași variabilă  $c$ , în ipoteza că:

- prima cifră citită de la tastatură este cea mai semnificativă cifră a lui  $x$ ;
- prima cifră citită de la tastatură este cea mai puțin semnificativă cifră a lui  $x$ .

58. Să se calculeze cu o precizie **eps** dată limita sirului  $x_k = \sum_{k=1}^{\infty} k * a^k$  știind că pentru  $|a| < 1$  sirul este convergent.

59. Pentru  $n$  cunoscut, să se calculeze suma:

$$S = \frac{1}{2} + \frac{1*3}{2*4} + \dots + \frac{1*3*...*(2n-1)}{2*4*...*(2n)}$$

**60.** Să se scrie un program care, folosind valoarea unui unghi  $x$  și o precizie  $\text{eps}$  dată ( $\text{eps} > 0$ ), calculează valoarea funcției  $\sin(x)$  cu o precizie (relativă)  $\text{eps}$ , folosind dezvoltarea în serie:

$$\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$$

cu  $x$  număr de tip real.

**61.** Să se scrie un program care să convertească o temperatură dată în grade C în echivalentul ei în grade Farenheit. Formula de conversie este:  $F=32+9/5C$ . Programul trebuie să tipărească ambele valori ale temperaturii cu mesajele corespunzătoare.

**62.** Scrieți un program care citește un număr natural  $n$  și determină, aplicând criteriile de divizibilitate, dacă este divizibil cu 2, 3, 4, 5, 9, 25. Se recomandă folosirea instrucțiunii *switch*.

**63.** Se dă un număr întreg  $n > 0$ .

- Găsiți toate perechile de numere întregi  $(x,y)$  pentru care  $x^2 + n = y^2$ .
- Găsiți perechea  $(a,b)$  pentru care  $n = (2^a)^*(2^b + 1)$ .

**64.** Să se genereze numărul natural  $n$ , pornind de la cifra 4 și efectuând la fiecare pas una din următoarele operații:

- împărțire la 2
- adăugarea la dreapta a cifrei 0
- adăugarea la dreapta a cifrei 4

Poate fi generat astfel orice număr natural  $n$ ?

**65.** Fiind dat un număr întreg  $n > 0$ , să se afișeze toate numerele naturale formate cu cifrele sale.

**66.** Scrieți o macro-definiție care să reprezinte cea mai mică dintre două valori numerice. Folosiți această macro-definiție în cadrul unui program.

67. Scrieți o macro-definiție care să reprezinte cea mai mare dintre trei valori numerice. Testați această macro-definiție cu ajutorul unui mic program.
68. Scrieți o macro-definiție care să furnizeze o valoare diferită de zero în cazul în care un caracter este literă mare (majusculă).
69. Scrieți o macro-definiție care să furnizeze o valoare diferită de zero în cazul în care un caracter este literă. Această macro-definiție va folosi alte două macro-definiții care să testeze faptul că este vorba de literă mică sau respectiv literă mare.
70. Scrieți o macro-definiție care să furnizeze o valoare diferită de zero în cazul în care un caracter este cifră ('0', '1', ..., '9'). Folosiți-o apoi pentru scrierea unei alte macro-definiții care să aibă ca rezultat o valoare diferită de zero în cazul în care caracterul testat este caracter special (adică nu este literă și nu este cifră). Se poate utiliza și macro-definiția din problema anterioară.
71. Scrieți o macro-definiție care să calculeze valoarea absolută a argumentului său. Asigurați-vă că o expresie de tipul *VALOARE\_ABSOLUT~(x + delta)* va fi corect evaluată.
72. Realizați un program care să afișeze succesorul unui număr memorat într-o variabilă de tip char. Testați programul inclusiv pentru valorile -1 și 128. Ce observați?
73. Să se scrie un program care convertește o valoare număr natural, introdusă de la tastatură, reprezentând numărul de secunde în ore, minute și secunde astfel încât să avem număr maxim de ore, respectiv număr maxim de minute. Formatul de afișare va fi:

<ore>H:<minute>M:<secunde>S

*Exemplu:* pentru un interval de 8358 secunde avem: 2H:19M:18S.

74. Scrieți un program care citește un număr întreg zecimal de cel puțin 2 cifre de la tastatură și îl afișează încadrat de 2 caractere #:  
a) în octal, pe 9 poziții, aliniat la stânga;  
b) în hexazecimal, pe 12 poziții, aliniat la dreapta;

75. Evaluarea la o disciplină din programa de învățământ se face în funcție de anumite ponderi pentru fiecare criteriu stabilit de profesor. Astfel, se vor citi date de la tastatură în următoarea ordine: nota la lucrarea scrisă, ponderea în procente corespunzătoare lucrării scrise, nota la laborator, ponderea notei de laborator în procente, nota la oral, ponderea în cadrul notei finale a notei de la oral, de asemenea, tot în procente. Scrieți un program care verifică dacă suma ponderilor introduse este egală cu 100%, iar dacă această condiție se verifică, se calculează nota elevului cu două zecimale după formula următoare:

$$(\text{pondere\_scris} * \text{nota\_scris} + \text{pondere\_oral} * \text{nota\_oral} + \text{pondere\_laborator} * \text{nota\_laborator}) / 100$$

În cazul în care suma ponderilor nu este 100%, se va afișa un mesaj de eroare corespunzător.

76. Se consideră funcția lui Collatz, dată prin următoarea expresie, unde n este un număr natural:

$$f(n) = \begin{cases} n/2 & \text{daca } n \text{ par} \\ 3*n+1 & \text{daca } n \text{ impar} \end{cases}$$

- a) Concepți un program care citește de pe mediul standard de intrare două numere naturale n, respectiv k, iar apoi calculați expresia:

$$E = ( f \circ f \circ \dots \circ f ) (n)$$

'o' de k ori

- b) Scrieți un program care citește un număr natural n, reprezentând argumentul funcției lui Collatz, și determină pe k din expresia anterioară astfel încât aceasta să aibă valoarea 1.

**77.** Se definește **Seria Farey de ordinul n** ca fiind sirul crescător al fracțiilor reduse, cuprinse în intervalul  $[0, 1]$ , în care numitorul este mai mic sau egal cu n. Termenii seriei Farey se notează cu  $(X_0, Y_0); (X_1, Y_1); (X_2, Y_2); \dots; (X_k, Y_k)$ ; ... Termenii  $X_k$  reprezintă numărătorii, iar  $Y_k$  numitorii.

Scrieți un program care generează **seria Farey de ordinul n**, pentru n citit de pe mediul standard de intrare. Afisarea seriei generate se va efectua conform convenției de notație prezentate în exemplul următor.

*Exemplu:* Seria Farey de ordinul 7 este următoarea:

$0/1, 1/7, 1/6, 1/5, 1/4, 2/7, 1/3, 2/5, 3/7, 1/2, 4/7, 3/5, 2/3, 5/7, 3/4, 4/5, 5/6, 6/7$

Reprezentarea acestei serii, conform convenției de notație:

$(0,1); (1,7); (1,6); (1,5); (1,4); (2,7); (1,3); (2,5); (3,7); (1,2); (4,7); (3,5); (2,3); (5,7); (3,4); (4,5); (5,6); (6,7); (1,1)$ .

Indicație:

Se observă că pentru orice număr natural n, avem relațiile:  $X_0=0$ ,  $Y_0=1$ ,  $X_1=i$ ,  $Y_1=n$ . Apoi, aplicând relațiile de recurență următoare până când rezultă valoarea 1 pentru ambele variabile, obținem termenii seriei Farey:  $X_{k+2} = (Y_{k+n} / Y_{k+1}) * X_{k+1} - X_k$ .

$$Y_{k+2} = (Y_{k+n} / Y_{k+1}) * Y_{k+1} - Y_k.$$

**78.** Se citește de pe mediul standard de intrare un număr natural n. Să se afișeze toate numerele naturale formate cu cifrele sale distințe.

*Exemplu:* Pentru n=8232 se vor afișa:

2 3 8 23 32 28 82 38 83 238 283 328 382 823 832

**79.** Se citesc de pe mediul standard de intrare două numere întregi. Scrieți un program care va afișa pe dispozitivul standard de ieșire "multiple" dacă unul din numere este multiplul celuilalt, respectiv "nemultiple", în caz contrar.

*Exemplu:*  $a = 32 \ b=9 \rightarrow$  nemultiple

$a=40 \ b=8 \rightarrow$  multiple

**80.** De pe mediul standard de intrare se introduc trei numere naturale reprezentând ziua (maxim două cifre), luna (maxim două cifre) și anul exact patru cifre. Să se verifice dacă cele trei numere formează o dată validă, iar în caz afirmativ se va afișa pe mediul standard de ieșire în formatul: zz/MM/AA, unde zz reprezintă ziua pe două cifre, MM luna pe două cifre, iar AA anul pe două cifre. În cazul în care nu se respectă condiția de dată validă, se va afișa un mesaj de eroare corespunzător.

*Exemplu:* pentru ziua=1, luna=9, anul=2003, se va afișa 01/09/03.

**81.** Se citesc de pe mediul standard de intrare două intervale de timp exprimate în ore, minute, secunde. Scrieți un program care verifică corectitudinea celor două intervale (reprezentate fiecare prin trei numere naturale), iar apoi calculează suma, respectiv diferența lor exprimând rezultatul în același mod. Se dorește ca suma, respectiv diferența să exprime numărul maxim de ore și minute în această ordine.

*Exemplu:* Un interval de timp rezultat de forma 26H, 90M, 70S nu este valid. Varianta corectă este : 27H, 31M, 10S.

**82.** Se citesc de pe mediul standard de intrare două intervale de timp exprimate prin zile, ore, minute, secunde și întoarce diferența dintre ele exprimată în secunde. Se are în vedere faptul că diferența dintre cele două intervale de timp poate depăși valoarea maximă pentru tipul de date long int.

**83.** Se citesc de pe mediul standard de intrare două intervale de timp exprimate prin zile, ore, minute, secunde și întoarce diferența dintre ele exprimată în același mod.

**84.** Se citește de pe mediul standard de intrare un număr natural n. Scrieți un program care va calcula următoarea sumă:  $S = \sum_{k=1}^n (-1)^k * k!$

**85.** Se citește de pe mediul standard de intrare un număr natural n. Să se afișeze toate valorile pozitive, supraunitare, puteri ale lui 2 mai mici decât n.

86. Scrieți un program care să afișeze codul ASCII al unui caracter citit de pe mediul standard de intrare în următorul format:

Format zecimal	Format octal	Format hexazecimal	Format binar	Caracterul
-------------------	--------------	-----------------------	--------------	------------

*Exemplu:*

032	040	020	00100000	SPATIU
125	175	07d	01111101	}

87. Scrieți un program care citește un număr real  $x$  și un număr natural  $n$  și calculează valoarea următorului polinom de ordin  $n$  în punctul  $x$ :

$$H_n(x) = 2 * x * H_{n-1}(x) - 2 * (n-1)H_{n-2}(x)$$
$$H_0(x) = 1; \quad H_1(x) = 2 * x$$

88. Se citește o cifră  $k$ . Să se afișeze toate numerele naturale cuprinse între 1 și 1000 al căror pătrat perfect se termină cu cifra  $k$ .

Observație: Se va afișa un mesaj în cazul în care se va introduce pentru  $k$  o valoare mică decât 0 sau mai mare ca 9.

89. Numerele naturale pot fi clasificate în: *deficiente*, *perfecte* și *abundente* după cum suma divizorilor săi este mai mică, egală sau mai mare decât valoarea numărului.

*Exemplu:*  $n = 12$  este abundant deoarece

$$S_d(12) = 1 + 2 + 3 + 4 + 6 = 16 > 12,$$

$$n = 6 \text{ este perfect deoarece } S_d(6) = 1 + 2 + 3 = 6$$

$$\text{iar } n = 14 \text{ este deficient deoarece } S_d(14) = 1 + 2 + 7 = 9 < 14.$$

90. Să se scrie un program care citește o valoare întreagă și specifică tipul numărului.

Observație: Se va afișa un mesaj de eroare în cazul în care se va introduce o valoare mai mică decât 0.

**91.** Se citește un număr  $k$ , natural. Să se afișeze toate numerele naturale cuprinse într-un interval de numere dat  $(a, b)$  care sunt divizibile cu  $k$ .  
Observație: Se va afișa un mesaj de eroare în cazul în care se va introduce pentru  $k$  o valoare negativă.

**92.** Să se genereze toate perechile de numere naturale  $(a,b)$ ,  $a \leq b$ , numerele ce formează o pereche având proprietatea că nu au nici o cifră comună și suma lor este egală cu un s dat.

**93.** Să se determine suma tuturor resturilor împărțirii la 25 pentru un număr de trei cifre.

**94.** Se citește un număr natural  $n$ . Să se scrie un program care determină cel mai mic numar natural cu  $n$  divizori.

*Exemplu:* pentru  $n=4$ , se va afișa 6.

**95.** Se citește un număr natural  $n$ . Să se afișeze toate numerele prime obținute din cifrele lui  $n$  prin permutări circulare.

*Exemplu:* pentru  $n=124$ , permutările 412, 241, 124, din care numai 241 este prim.

**96.** Se citește un număr natural  $n$ . Să se afișeze toate numerele  $k \in \{2, 3, \dots, n-1\}$  care sunt prime cu  $n$ .

*Exemplu:* pentru  $n=8$ , avem numerele 3, 5, 7, care vor fi afișate.

**97.** Se dă un număr real cu cel mult 7 zecimale. Să se determine diferența patratelor cifrelor ce constituie partea sa întreagă și suma patratelor cifrelor zecimale.

*Exemplu:*  $x=12345678.377259 \rightarrow 102 - 219 = -117$

**98.** Scrieți un program care verifică **conjunctura lui Goldbach**. oonjunctura constă în posibilitatea scrierii oricărui număr par mai mare 2 ca

sumă a două numere prime. Programul va citi de la tastatură numerele m și n, afișând câte o conjunctură pentru fiecare număr cuprins între m și n.

*Exemplu:* Pentru m=700 și n=800 se va afișa:

$$700 = 683 + 17$$

$$702 = 691 + 11$$

$$704 = 701 + 3$$

.....

$$800 = 793 + 7$$

**99.** Scrieți un program care calculează pătratul unui număr cu maxim 4000 de cifre, care începe cu cifra 3 și următoarele cifre sunt toate 5.

*Exemplu:*  $3555^2 = 12638025$

**100.** Scrieți un program care realizează următoarea înmulțire criptată, în care se înlocuiesc caracterele \* cu cifre de la 0 la 9:

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

Observație: Prima cifră a oricărui produs parțial, din rezultatul final, a înmulțitorului sau a deînmulțitului nu poate fi zero. În plus, cele 10 cifre din înmulțit, înmulțitor și din rezultat să fie distințe două cîte două, adică să utilizeze toate cele 10 cifre 0, 1, 2, ..., 9 în cele trei numere.

## 1.6. Teste grilă



1.6.1. Care dintre valorile de mai jos sunt constante întregi scrise corect?

- a) 123      b) -17      c) +843      d) 0154      e) --67



1.6.2. Care dintre construcțiile de mai jos reprezintă constante caracter?

- 1)" "    2)\'    3)'a'    4)' "'    5)\'\|'    6)\'13'    7)"a"    8)' '
- a) 2), 3) și 8)      b) toate      c) toate mai puțin 5) și 6)  
d) 2), 3), 4) și 8)      e) 3), 4) 5), 6) și 8)



1.6.3. Pentru fiecare dintre constantele aflate în coloana A), alegeti din coloana B) tipul său:

Coloana A)	Coloana B)
A1) 5.0	B1) constantă întreagă
A2) 5	B2) constantă reală
A3) '5'	B3) constantă hexazecimală
A4) 05	B4) constantă octală
A5) "05"	B5) constantă caracter
A6) 0x5	B6) constantă sir de caractere

- a) A1 → B2, A2 → B1, A3 → B5, A4 → B1, A5 → B6, A6 → B3  
b) A1 → B2, A2 → B1, A3 → B5, A4 → B4, A5 → B5, A6 → B3  
c) A1 → B2, A2 → B1, A3 → B5, A4 → B4, A5 → B6, A6 → B3  
d) A1 → B2, A2 → B1, A3 → B5, A4 → B4, A5 → B6, A6 → eronată  
e) A1 → B2, A2 → B1, A3 → B5, A4 → B1, A5 → B6, A6 → eronată



1.6.4. Care dintre următoarele declarații de variabile declară corect o variabilă x ce poate memora valori reale?

- a) float x;      b) double x;      c) unsigned float x;      d) x:float;      e) x:double;



1.6.5. Care dintre liniile de program de mai jos realizează inițializarea corectă a variabilei x la declararea sa?

- a) int x==2;      b) x:int=2;      c) int x=2;      d) int x 2;      e) x=2 : int;



1.6.6. Care dintre variabile vor avea valori întregi după execuția sevenței de program următoare?

```
int a=3,b,c;  
float x=-11.23;  
char d;  
b=x; d='A'; cs'M'-'N';
```

- a) variabila x    b) variabila c    c) variabila d    d) variabila a.    e) variabila b



1.6.7. Definiți o constantă simbolică PI cu valoarea 3.14, folosind directiva-preprocesor "#def ine".

- a) #define 3.14 PI;
- b) #define PI 3.14;
- c) #define float PI 3.14;
- d) #define PI=3.14;
- e) #define float PI=3.14;



1.6.8. Care dintre programele de mai jos nu conțin erori și afișeză cuvintele "Program" și "simplu" unul sub altul (fiecare pe câte un rând) ?

<b>a)</b> <pre>#include&lt;iostream.h&gt; int main() ; {     cout &lt;&lt; "Program";     cout &lt;&lt; "\n" &lt;&lt; "simplu"; }</pre>	<b>b)</b> <pre>#include&lt;iostream.h&gt; main() {     cout &lt;&lt; "Program\n";     cout &lt;&lt; "simplu"; }</pre>
--	--

c) <pre>#include&lt;iostream.h&gt; int main() {     cout &lt;&lt; "Program\nsimplu";     cout &lt;&lt; "\n" ; }</pre>	d) <pre>#include&lt;iostream.h&gt; int main() {     cout &lt;&lt; "Program";     cout &lt;&lt; "simplu" &lt;&lt; "\n"; }</pre>
--	---

e) nici unul dintre programele anterioare



1.6.9. Ce valoare afisează programul următor?

- a) -7
- b) -8
- c) -9
- d) 4
- e) 5

```
#include <iostream.h>
#include <math.h>
int main()
{
    int x=4, y=-9, z=(x+y)/2, u;
    z--;
    u=(sqrt(x)) + sqrt(abs(y)) / (x+y);
    z=u%2-x-y;
    cout << "\n" << z;
}
```



1.6.10. Fie variabilele întregi  $a=1$ ,  $b=2$ ,  $c=3$ ,  $d=4$ . Care dintre construcțiile de mai jos sunt expresii scrise corect, cu valoarea 0?

- a) !d
- b) a+d<d
- c) a\*b+c
- d) a=b<c
- e) (a<b) != (b<c)



1.6.11. Care dintre următoarele afirmații sunt adevărate?

- a) Operatorul de atribuire este "=".
- b) Operatorul care realizează "SAU logic" între două expresii este "&&".
- c) " != " este un operator logic.
- d)  $a \% b$  reprezintă restul împărțirii întregi a lui  $a$  la  $b$ .
- e) Într-o expresie, operatorii relaționali se execută înaintea celor aritmetici.



1.6.12. Pentru care dintre seturile de valori ale variabilelor  $x$ ,  $y$ ,  $z$  de mai jos expresia  $(x < y) < (z != x) < (z - y) < x$  are valoarea 1?

- a)  $x=3; y=5; z=4$       b)  $x=4; y=3; z=4$       c)  $x=3; y=4; z=3$   
 d)  $x=5; y=4; z=3$       e)  $x=5; y=5; z=5$



1.6.13. Fie variabilele intregi  $a$  si  $b$ . Expresia se scrie in C++ astfel:

$$\frac{a \cdot b + \frac{a - b}{2}}{\frac{a - 2}{b - 2} - \frac{a}{b}}$$

- a)  $((a*b)+(a-b)/2)/((a-2)/(b-2)-(a/b))$       b)  $(a*b+(a-b)/2)/((a-2)/(b-2)-a/b)$   
 c)  $(a*b+(a-b)/2)/a-2/b-2-(a/b)$       d)  $((a*b+(a-b)/2)/a-2/b-2)-a/b$   
 e)  $(a*b+(a-b)/2)/(a-2/b-2-a/b)$



1.6.14. Fiind date variabilele întregi  $a$  și  $b$ , cum se scrie corect condiția "a mai mic decât b și b mai mic decât c"?

- a)  $(a < b) \&\& (b < c)$       b)  $(a < b) \&(b < c)$   
 c)  $a < b \&\& b < c$       d)  $(a < b) \parallel (b < c)$   
 e)  $a < b \mid b < c$



1.6.15. Care dintre următoarele expresii au valoarea 1 dacă și numai dacă valorile variabilelor întregi  $x$  și  $y$  sunt numere pare?

- a)  $x-y==2$       b)  $x*y \% 4 == 0$   
 c)  $(x+y)\%2==0$       d)  $y \% x == 2$   
 e)  $(x \% 2 == 0) \parallel (y \% 2 == 0)$



1.6.16. Care dintre următoarele expresii sunt adevărate dacă și numai dacă valorile variabilelor  $x$  și  $y$  sunt numere naturale consecutive?

- a)  $x-y==1$       b)  $(x==1) \&\& (y==2)$   
 c)  $(x-y==1) \&\& (y-x==1)$       d)  $y==x \pm 1$   
 e)  $(x-y==1) \parallel (y-x==1)$



1.6.17. Se consideră următoarele declarații de variabile:

int a,b,e; float c,d;

Care dintre instrucțiunile de mai jos sunt incorecte?

a)  $a=a*b;$

b)  $e=a<c;$

c)  $-b=a+a/b;$

d)  $\text{cout} << (d=(a+b)/2);$

e)  $c*d=a-b;$



1.6.18. Fie variabilele a, b, c și m toate de tipul float. Care dintre sevențele de mai jos afișează corect media aritmetică a numerelor a, b și c?

a)  $\text{printf}("%f", (a+b+c)/3);$

b)  $\text{printf}("%f", a+b+c/3);$

c)  $\text{printf}("%d", a/3+b/3+c/3);$

d)  $\text{printf}("%f", ((a+b)/2+c)/2);$

e)  $\text{printf}("%f", m=(a+b+c)/3);$



1.6.19. Fie variabilele x, y și u de tipul int. Care dintre instrucțiunile de mai jos mărește valoarea variabilei u cu câtul întreg al împărțirii lui x la y?

a)  $u+-x\%y;$

b)  $u=x\%y+u;$

c)  $u=x/y;$

d)  $u+=x/y;$

e)  $u=x/y+u;$



1.6.20. Fie variabilele a, b, c de tipul int. Care dintre următoarele echivalențe între expresii sunt adevărate? (două expresii se consideră echivalente dacă produc același rezultat în orice situație):

a) Expresia  $!(a!=b)$  este echivalentă cu  $!a!=b$

b) Expresia  $a= (b < c)$  este echivalentă cu  $a=b < c$

c) Expresia  $a \&\& b$  este echivalentă cu  $a==0 \&\& b=0$

d) Expresia  $(a>c) \&\& (b>c)$  este echivalentă cu  $a>c \&\& b>c$

e) Expresia  $!(a>=b \mid\mid a>=c)$  este echivalentă cu  $a < b \&\& a < c$



1.6.21. Fie variabilele x, y, z de tipul int, fiind cunoscute valorile inițiale x=3, y=5 Care dintre instrucțiunile de mai jos trebuie executată astfel, încât, după execuție, valoarea variabilei z să fie 21?

- a)  $z=2*x+3*y--;$
- b)  $z=2*x+3*-y;$
- c)  $z=2*x---+3*y;$
- d)  $z=2*---x+3*y;$
- e)  $z=2*x+3*y;$



1.6.22. Pentru programul următor, precizați care dintre afirmațiile de mai jos sunt adevărate:

```
#include <stdio.h>
int main ()
{
    int a,b;
    float c;
    scanf("%d %d", &a, &b);
    printf("%d\n",!a);                                // (1)
    printf("%d\n",b%2==0 && b>0);                  // (2)
    printf("%-8.2f\n",float(2*b+a));                // (3)
    printf("a=%3d$b=%-3d\n",a,b);                   // (4)
}
```

- a) Dacă valoarea citită în variabila a este diferită de zero, atunci linia (1) afișează valoarea 0
- b) Dacă valoarea citită în variabila b este pară și pozitivă, atunci linia (2) afișează valoarea 0
- c) Instrucțiunea (3) afișează corect valoarea expresiei  $2*b+a$ , pe opt caractere din care două zecimale, cu aliniere la dreapta
- d) Dacă de la tastatură se introduc valorile 2 și 6 pentru variabilele a, respectiv, b, atunci linia (4) va afișa

$a=2\bullet\bullet \$b=\bullet\bullet 6$

(prin "•" am simbolizat caracterul "spațiu")

- e) Nici una dintre afirmațiile anterioare nu este adevarată, deoarece citirea cu funcția `scanf` este eronată



1.6.23. Funcțiile `getchar()`, `getch()` și `getche()` citesc de la tastatură un caracter. Ce deosebiri există între cele trei funcții?

- a) Funcțiile getchar() și getche() realizează citirea cu ecou, iar getch() citește caracterul fără ecou
- b) Funcția getchar() citește caracterul cu ecou, iar funcțiile getche() și getch() realizează citirea fără ecou
- c) Funcțiile getchar() și getch() preiau caracterul numai după apăsarea tastei ENTER
- d) Funcțiile getchar() și getche() preiau caracterul de îndată ce a fost tastat, fără să mai aștepte "confirmarea" cu ENTER
- e) Toate cele trei funcții au prototipul în header-ul conio.h



**1.6.24.** Fie trei variabile întregi a, b, x. Scrieți cu ajutorul unei expresii condiționale enunțul: "daca  $x \notin [a, b]$  atunci x ia valoarea lui a, în caz contrar x ia valoarea lui b".

- a)  $x=((x < a) || (x > b)) ? a : b;$
- b)  $x= <x < a || x > b) ? a : b;$
- c)  $x- ((x < a) \&& (x > b)) ? a : b;$
- d)  $x= (x < a) | | (x > b) ? b : a ;$
- e)  $((x < a) || (x > b)) ? (x=a):(x=b);$



**1.6.25.** Știind că în standardul ASCII caracterele literă mare au codurile succesive începând cu 65 ('A'←65, 'B'←66, 'C'←67, etc), deduceți ce valoare va afișa programul următor.

- a) 1
- b) 3
- c) 69
- d ) 67
- e) 0

```
#include<stdio.h>
int main()
{
    int x,y,z,p;
    char n,m;
    m='C';      n='A';
    x=m;  y=2*m-n;   z=3;
    p=x<y ? (y<z ? z : y) : (z<x ? x : z);
    printf("\n%d",p);
}
```



1.6.26. Presupunem ca rulăm programul următor sub o versiune a limbajului C++, în care valorile de tipul int se memorează pe doi octeți, iar cele de tipul float pe patru octeți. De câte ori va afișa programul valoarea 2?

- a) nici o dată
- b) o dată
- c) de două ori
- d) de trei ori
- e) de patru ori

```
#include<iostream.h>
#include<stdio.h>
int main()
{
    int x; char c;
    cout<<"\n";
    x='A'; cout << sizeof(x);
    c='A'; printf("%d", sizeof(c));
    printf("%d", sizeof(float)-2);
    x=sizeof(int);
    x=++x/2;
    cout<< (x==2);
}
```



1.6.27. Ce valori afișează programul următor?

- a) 10 18 16
- b) 11 18 18
- c) 10 18 18
- d) 11 18 17
- e) 10 18 17

```
#include<stdio.h>
int main()
{
    int x=10, y=6, m,n,p;
    n=(m=x++, y++, p=x+y);
    printf("\n%d %d %d",m,n,p);
}
```



1.6.28. Ce valoare putem introduce la citirea variabilei y, astfel încât programul de mai jos să tipărească 1?

- a) 2
- b) 3
- c) 4
- d) orice valoare pară
- e) orice valoare impară

```
#include<iostream.h>
int main()
{
    int x=2, y, z;
    cin >> y;
    z=y+3*x++;
    cout<< "\n" << ((z%2==0 && x>=1) ? 1 : 0);
}
```



1.6.29. În programul următor, care dintre secvențele de instrucțiuni notate cu (1), (2) și (3) nu vor produce erori la execuție?

```
#include <stdio.h>
int main()
{
    int x,y;
    char m;
    m='A';
    putchar(m+1); // (1)
    x=getchar(); m='A'; x=m; printf("%c %d ",x,x); // (2)
    x=300; m=x; y=m; printf("\n%c %d\n",y,y); // (3)
}
```

- a) Numai secvențele (1) și (3)
- b) Toate trei
- c) Numai secvența (1)
- d) Numai secvențele (1) și (2)
- e) Nici una



1.6.30. Fie variabilele a, b, c, d, toate de tipul int, cu valorile a=1, b=2, c=3 și d=4. Care dintre instrucțiunile de mai jos poate fi executată, astfel încât după execuție, noile valori ale variabilelor să fie a=4, b=3, c=3 și d=4?

- a) a = b = c = d;
- b) a = (b = c) = d;
- c) (a = (b = c)) = d;
- d) a = ((b = c) = d);
- e) (a = b = c) = d;



1.6.31. Precizați valoarea lui n, rezultată în urma execuției programului:

- a) Programul este greșit
- b) 97
- c) 99
- d) 79
- e) NULL

```
#include<iostream.h>
int main()
{
    char c; int
    n=97;
    n=c=n;
}
```



1.6.32. Precizati valoarea pe care o va avea variabila c în urma executiei programului de mai jos:

- a) 'd'
- b) 'c'
- c) 'b'
- d) NULL
- e) atribuirea este greșită

```
#include<iostream.h>
int main()
{
    char c='d';
    int n=99;
    c=n+1=c-1;
}
```



1.6.33. Fie variabilele a, b, c, de tipul int, cu valorile a=11, b=5, c=7. Care dintre expresiile de mai jos are valoarea 1?

- a)  $(a \mid \sim b) \& 1$
- b)  $(\sim a \& b) \mid 1$
- c)  $(a \& \sim b) \mid 1$
- d)  $(a \& b) \mid \sim 1$
- e)  $(\sim a \mid b) \& 1$



1.6.34. Care dintre următoarele secvențe de instrucțiuni atribuie variabilei întregi x cea mai mică dintre valorile variabilelor întregi a și b, sau valoarea lor comună în cazul în care acestea sunt egale ?

- a) if (a=b) x=b;
- b) x=a;
- else
- if (x>b)
- x=b;
- if (b>a) x=a;
- c) if (b>=a) x=a;
- d) if (a<=b) x=b;
- else x=b;
- else x=a,
- e) Nici una dintre secvențele anterioare.



1.6.35. Precizați de câte ori se va afișa valoarea 1 în timpul execuției programului următor, dacă prin citire de la tastatură variabilele primesc valorile a=3, b=4 și x=5.

- a) nici o dată
- b) o dată
- c) de două ori
- d) de trei ori
- e) de patru ori

```
#include<iostream.h>
int main()
{
    int a, b, x;
    cin>>a>>b>>x;
    if (!(x<=a) && (x>=b)) cout<<1<<"\n";
    if (!(x<=a || x>=b)) cout <<1<<"\n";
    if (!(x<=a) && !(x>=b)) cout<<1<<"\n";
    if (!(x<=a) || !(x>=b)) cout <<1<<"\n";
}
```



1.6.36. Precizați ce afișează programul de mai jos:

```
#include <iostream.h>
int main ()
{
    int a,b,c;
    a = c = 1; b = 0;
    if (a || b)
        if (b && !c) cout<<"unu";
        else cout << "doi";
}
```

- a) Nu afișează nimic
- b) Textul unu
- c) Textul doi
- d) Instrucțiunea if este greșit sintactic
- e) Nici una dintre afirmațiile de mai sus



1.6.37. În urma execuției secvenței de program alăturate, pentru care dintre perechile de valori ale variabilelor întregi a și b date mai jos se va afisa x=5?

- a) a=-1 și b=-1
- b) a=1 și b=1
- c) a=1 și b=-1
- d) a=0 și b=-1
- e) a=1 și b=0

```
if (a>0)
if (b>0) x=8;
else x=5 ;
else x=6;
cout << x;
```



1.6.38. Dacă în timpul execuției programului de mai jos n va primi prin citire de la tastatură valoarea 232213, care vor fi în final valorile variabilelor f1, f2 și f3?

- a) f1=1, f2=1, f3=1
- b) f1=1, f2=2, f3=2
- c) f1=1, f2=3, f3=2
- d) f1=2, f2=1, f3=3
- e) f1=3, f2=2, f3=1

```
#include<iostream.h>
int main()
{
    long n;
    unsigned int f1,f2,f3,c;
    cin >> n;
    f1=f2=f3=0;
    do{
        c=n%10;
        n=n/10;
        switch(c)
        {
            case 1: {f1++; break;}
            case 2: {f2++; break;}
            case 3: {f3++; break;}
        }
    }while(n!=0);
    cout << f1 << f2 << f3;
}
```



1.6.39. Pentru n=7, care dintre secvențele de program de mai jos trebuie executată astfel, încât, la finele execuției, valoarea variabilei P să fie 48?

a)  
P=1 ; i=2 ;  
while (i<=n)  
{  
 P\*=i ; i+=2 ;  
}

b)  
P=1; i=1;  
while (i<n/2)  
{  
 i++;  
 P=P\* (2\*i+1) ;  
}

c)  
P=1; i=1;  
while (i<=n/2)  
{  
 P=P\* (2\*i);  
 i++;  
}

d)  
P=1; i= 0;  
while (i<n)  
{  
 i+=2;  
 P\*=i ;  
}

e) Nici una dintre secvențele anterioare.



1.6.40. Precizați care dintre următoarele secvențe de instrucțiuni atribuie variabilei întregi  $x$  valoarea  $n^2$ , cu  $n$  număr natural, variabila auxiliară  $i$  fiind de tip întreg.

- a)  $x=1;$   
`for(i=1;i<3;i++) x*=n ;`
- b)  $x=1;$   
`for(i=1;i<=n;i++) x*=2;`
- c)  $x=1; i=0;$   
`while (i<2) x*=n; i++;`
- d)  $x=1; i=0;$   
`do { i++; x*=n;`  
`} while (i<2);`
- e)  $x = x*n;$



1.6.41. Precizați care dintre următoarele secvențe de instrucțiuni atribuie variabilei întregi  $x$  valoarea  $10^n$ , cu  $n$  număr natural, variabila auxiliară  $i$  fiind de tip întreg.

- a)  $x=10;$   
`for (i=1; i<=n; i++)`  
`x*=i;`
- b)  $x=1;$   
`for (i=n; i>0; i--)`  
`x*=10;`
- c)  $x=1; i=1;$   
`do { x*=10;`  
`i++;`  
`} while (i<n);`
- d)  $x=1; i=0;$   
`while (i<=n) {`  
`i++;`  
`x*=i;`  
`}`
- e) Nici una dintre variantele anterioare.



1.6.42. Deducreți ce valoare se va afișa în urma execuției secvenței de program de mai jos, dacă valorile variabilei  $x$  citite de la tastatură sunt în ordine 3,2,4,3,5,10,20,0.

- a) 0
- b) 1
- c) 2
- d) 3
- e) 4

```

    cin >> x;
    nr = 0;
    do{
        y=x;
        cin >> x;
        if(x==2*y)
            nr++;
    } while(x!=0);
    cout << nr;

```



**1.6.43.** Care va fi valoarea variabilei c afișată de către programul următor, dacă de la tastatură se citesc valorile a=b=3?

- a) 8
- b) 32
- c) 27
- d) 13
- e) 1

```

#include<iostream.h>
int main()
{
    long a,b,c,z,i;
    cout<<"Dati a si b: " ;
    cin>>a>>b ;
    for(i=1 ; i<=a ; i++)
    {
        c+=z ;
        z*=b ;
    }
    cout<< "c = "<<c ;
}

```



**1.6.44.** Care dintre secvențele de program  $S_1$ ,  $S_2$ ,  $S_3$ , date mai jos, este echivalentă cu secvența alăturată? (Două secvențe de program se consideră echivalente, dacă produc același efect în orice situație) Toate variabilele folosite sunt întregi.

```

p=1;
for (i=1; i<=n;i++)
{
    s=0;
    for (j=1; j<=i;j++) s=s+j;
    p=p*s;
}

```

```
// Secvența S1
p=1; s=0;
for(i=1;i<=n;i++)
{
    p=p*i;
    s=s+p;
}
```

```
// Secvența S2
p=1; s=0;
for(i=1;i<=n;i++)
{
    s=s+i;
    p=p*s;
}
```

```
// Secvența S3
p=1; s=0;
for(i=1;i<=n;i++)
{
    p=p*s;
    s=s+i;
}
```

- a) numai S1    b) numai S2    c) numai S3    d) toate    e) nici una



1.6.45. Care dintre următoarele secvențe de instrucțiuni atribuie variabilei u valoarea primei cifre a numărului natural reprezentat de variabila x?

- a)  $u=x;$   
**b) while ( $u \geq 10$ )  $u=u \% 10;$**   
c)  $u=x/10;$   
d)  $u=x \% 10;$   
e) Nici una dintre variantele anterioare



1.6.46. Care dintre următoarele secvențe de instrucțiuni atribuie variabilei u valoarea ultimei cifre a numărului natural reprezentat de variabila x?

- a) **while ( $x \geq 10$ )  $x=x/10;$**   
 $u=x;$   
c)  $u=x \% 10;$   
d)  $u=x \% 10;$   
e) Toate variantele anterioare.



1.6.47. Fie secvența de program următoare, în care ok este o variabilă de tipul int, iar x este un număr natural.

```
ok=0;
for(i=2; i<x; i++)
    if(x % i == 0) ok == 1;
cout<<ok;
```

Secvența afișează 1 dacă:

- a) Numărul x are cel puțin un divizor propriu
- b) Numărul x nu are nici un divizor propriu
- c) Toate numerele naturale mai mici decât n, fără 0 și 1, sunt divizori proprii ai lui x
- d) Numărul x are cel mult un divizor propriu
- e) Nici una dintre variantele de mai sus



**1.6.48.** Se consideră secvențele de program de mai jos. Pentru  $n=4$ , precizați care dintre secvențe afișează, în urma execuției, sirul de numere: 12233344444

**a)**

```
for(i=1;i<=n;i++)
    for
        (j=1;j<=n;j++)
            printf ("%2d", i);
```

**b)**

```
for (i=1;i<=n;i++)
    for (j=1;j<=i;j++)
        printf ("%2d",i);
```

**c)**

```
for (i=1 ;i<=n ;i++)
    for
        (j=1;j<=n;j++)
            printf ("%2d",i);
```

**d)**

```
for (i=1;i<=n;i++)
    for (j=1;j<=i;j++)
        printf ("%2d",j);
```

**e)**

```
for (j=1;j<=n;j++)
    for
        (i=1;i<=n;i++)
            printf ("%2d",i);
```



**1.6.49.** Pentru afișarea numerelor naturale  $1,2,\dots,n$  (unde  $n$  presupune cunoscut), propunem următoarele două secvențe de program:

*S1)*

```
for(i=1;i<=n;i++)
    printf ("%2d",i);
```

*S2)*

```
for(i=1;i<=n;i++);
    printf("%2d",i);
```

Care dintre afirmațiile de mai jos este adevarată?

- a) Nici una dintre secvențe nu îndeplinește cerința problemei
- b) Ambele secvențe îndeplinesc cerința problemei
- c) Numai secvența *S2*) conține erori de sintaxă
- d) Numai secvența *S1*) îndeplinește cerința problemei
- e) Numai secvența *S2*) îndeplinește cerința problemei



1.6.50. Fie secvența de program următoare:

```
s=0;  
for(i=3; i<=n; i+=3)  
    s+=i;
```

Se dau mai jos cinci triplete de numere, fiecare astfel de triplet reprezentând un set de valori pentru variabila de intrare n. Care dintre aceste triplete au proprietatea că pentru toate cele trei valori ale lui n din triplet se obține aceeași valoare a lui S ?

- a) (3, 5, 6)    b) (6, 7, 8)    c) (10, 11, 12)    d) (6, 9, 12)    e) (15, 16, 17)



1.6.51. Considerând că toate variabilele sunt întregi, ce valoare se afișează după execuția secvenței de mai jos:

- a) 1  
b) 5  
c) 6  
d) 51  
e) 63

```
s=0; t=0; x=3; i=1; y=1; z=1;  
do{  
    if(x>0)  
        if(y>1)  
            if(z>2) t=x;  
            else t=x+y;  
        else t=x+y+z;  
        s+=i+t;  
        i++;  
    }while(i>7);  
cout<<s;
```



1.6.52. Pentru ce valoare a variabilei M, secvența de program de mai jos reprezintă o buclă infinită?

- a) 10  
b) orice valoare diferită de 10  
c) 0  
d) orice valoare diferită de 0  
e) pentru orice valoare întreagă

```
int n=10, M;  
do {  
    while (n>0) n--;  
} while (n!=M);
```



**1.6.53.** Se dă programul de mai jos. Știind că prima valoare citită (cea a variabilei a) este 4, precizați ce valori trebuie citite pentru variabila b în corpul ciclului, astfel încât, în final, să se afișeze valoarea 4.

- a) 1, 2, 3, 4
- b) 2, 3, 4
- c) 1, 3, 4
- d) 1, 2, 4
- e) 4, 4, 4

```
#include <iostream.h>
int main()
{
    int n=1, a, b;
    cin >> a;
    do {
        cin >> b;
        n++;
    } while(b!=a);
    cout << n;
}
```



**1.6.54.** Definim oglinditul unui număr natural x ca fiind numărul obținut prin citirea numărului x în ordine inversă (de la dreapta la stânga).

*Exemplu:* Oglinditul numărului 1534 este 4351. Care dintre programele de mai jos afișează corect oglinditul lui x?

```
//Programul P1
#include <stdio.h>
int main()
{
    int c; long d,x,y;
    scanf("%ld",&x);
    d=x; y=0;
    while (d)
    {
        c=d % 10;
        y=y+c*10;
        d=d/10;
    }
    printf("%ld",y);
}
```

```
//Programul P2
#include <stdio.h>
int main()
{
    int c; long d,x,y;
    scanf("%ld",&x);
    d=x; y=0;
    do
    {
        c=d% 10;
        y=y+c*10;
        d=d/10;
    } while(d);
    printf("%ld",y);
}
```

```
// Programul P3
#include <stdio.h>
int main()
{
    int c; long d,x,y;
    scanf("%ld",&x);
    d=x; y=0;
    while (d)
    {
        c=d%10;
        y=y*10+c;
        d=d/10;
    }
    printf("%ld",y);
}
```

```
//Programul P4
#include <stdio.h>
void main()
{
    int c; long d,x,y;
    scanf("%ld",&x);
    d=x; y=0;
    do
    {
        c=d% 10;
        y=y*10+c;
        d=d/10;
    } while (d);
    printf("%ld",y);
}
```

- a) P1 și P2
- b) P1 și P3
- c) P2 și P4
- d) P1 și P4
- e) P3 și P4



1.6.55. Precizați ce valori se vor afișa, în ordine, în timpul execuției programului următor:

- a) 0,0,0
- b) 0,0,1
- c) 0,1,1
- d) 1,1,1
- e) 1,0,1

```
#include<iostream.h>
int main()
{
    int x,y,m,n,a,b=5;
    x=(m=b=n=3,b+4);
    y=a=((b=5) ? b-- : -b);
    if( (!y==a) && (m==n) ) cout <<1; else cout << 0;
    if( !a && b && !m ) cout << 1; else cout << 0;
    if( (n=(a+b--)) == (m-x) ) cout <<1; else cout <<0;
}
```



1.6.56. Care dintre sirurile de valori date în variantele de răspuns trebuie introduse de la tastatură în timpul execuției programului următor, astfel încât să se declanșeze un ciclu infinit ?

- a) 2,7,3,8,0,0  
 b) 2,5,4,4,0,0  
 c) 1,3,6,2,0,0  
 d) 2,4,5,8,0,0  
 e) 0,0

```
#include <stdio.h>
int main ()
{
    int x,y;
    while (scanf ("%d",&x)==1 && scanf ("%d",&y)==1 && (x||y))
        do {y--;
            printf ("%d %d",x,y);
        } while (x!=y);
}
```



1.6.57. Fie secvența de program următoare, în care:

- variabilele i, n și s sunt întregi
- <v1>, <v2>, <v3> simbolizează valori constante întregi
- <cond> este o condiție (expresie logică)

Alegeți de mai jos valorile <v1>, <v2>, <v3>, precum și condiția <cond>, astfel încât programul să afișeze suma cifrelor naturale 1+2+...+9.

```
n=<v1>; i=<v2>; S=<v3>;
while ( <cond> )
    S=S+i++;
    printf("%d",S);
```

- a) <v1> ← 10, <v2> ← i, <v3> ← 0, și condiția "i<n",  
 b) <v1> ← 10, <v2> ← 1, <v3> ← 0, și condiția "i<=n";  
 c) <v1> ← 10, <v2> ← 0, <v3> ← 0, și condiția "i<n";  
 d) <v1> ← 9, <v2> ← 1, <v3> ← 0, și condiția "i<n";  
 e) <v1> ← 9, <v2> ← 2, <v3> ← 1, și condiția "i<=n";



1.6.58. Se consideră următoarele două secvențe de instrucțiuni în care:

- toate variabilele sunt de tipul întreg
- <v1>, <v2>, <v3> simbolizează valori constante întregi

Determinați valorile <v1>, <v2>, <v3> astfel, încât, după execuția celor două secvențe diferența dintre valorile variabilelor y și x să fie 10 ( $y-x=10$ ).

```
//Secvența S1
i=<v1>; x=0;
while (i<=10)
{
    x+=10; i++;
}
```

```
//Secvența S2
j=<v2>; y=0;
do
{
    j++; y+=10;
} while (j < <v3>);
```

- a) <v1> ← 0, <v2> ← 0, <v3> ← 11
- b) <v1> ← 1, <v2> ← 0, <v3> ← 11
- c) <v1> ← 1, <v2> ← 0, <v3> ← 10
- d) <v1> ← 0, <v2> ← 0, <v3> ← 12
- e) <v1> ← 1, <v2> ← 1, <v3> ← 13



1.6.59. Ce valoare va afișa programul următor pentru n=12 ?

- a) 0
- b) 9
- c) 12
- d) 78

e) programul conține erori

```
#include <iostream.h>
int main ()
{
    int i,n,s ;
    cin >> n ;
    for(s=0, i=2; i<n/2; !(n%i) ? s+=i++ : i++) ;
        cout<< s;
}
```



1.6.60. Pentru programul următor, care dintre afirmațiile de mai jos sunt adevărate ?

```
#include <iostream.h>
int main()
{
    int S,x;
    for(S=0, x=1; 0; S+=x, cin >> x)
        if (!x) break;
    cout << S;
}
```

- a) Dacă de la tastatură se introduc, în ordine, numerele 2, 3, 4 și 5, atunci programul va afișa suma numerelor citite, adică 14.
- b) Dacă prima valoare introdusă de la tastatură este 0, atunci ciclul se încheie și se afișează valoarea 1.
- c) Ciclul este eronat: nu se poate face o citire în linia `for`.
- d) Instrucțiunea `if` este eronată.
- e) Din cauză că lipsește expresia care dă condiția de continuare, ciclul `for` se va executa la infinit.



**1.6.61.** Dacă de la tastatură se introduc, în ordine, numerele 2, 7, 3, 8, 5, 5, ce valoare va afișa secvența următoare?

- a) 0  
b) 1  
c) 2  
d) 3  
e) 4

```
int a,b, nr=0;
do {
    cin >> a >> b;
} while ((b!=a) ? ++nr : 0 );
cout << nr;
```



**1.6.62.** Care dintre secvențele de mai jos afișează corect sirul cifrelor zecimale impare 97531 în această ordine?

a)  
`for(i=9; i>=1; i--)`  
`cout << i--;`

b)  
`for(i=0; i<=9; i++)`  
`cout << 9-i--;`

c)  
`for(i=9; i-->=1;)`  
`cout << i,i--;`

d)  
`i=10;`  
`while (i--)`  
`cout << --i;`

e)  
`i=1;`  
`do {`  
 `cout << 10 - i++;`  
`} while ( i<=9 ? i++ : 0 );`

## 1.7. Răspunsuri la întrebările grilă

- 1.6.1. a), b), c), d)
- 1.6.2. e)
- 1.6.3. c)
- 1.6.4. a) și b)
- 1.6.5. c)
- 1.6. b), c), d) și e)
- 1.6.7. b)
- 1.6.8. b) și c)
- 1.6.9. a)
- 1.6.10. a) și e)
- 1.6.11. d)
- 1.6.12. b) și e)
- 1.6.13. a) și b)
- 1.6.14. a) și c)
- 1.6.15. e)
- 1.6.16. e)
- 1.6.17. c) și e)
- 1.6.18. a) și e)
- 1.6.19. d) și e)
- 1.6.20. b), d) și e)
- 1.6.21. a), c) și e)
- 1.6.22. a)
- 1.6.23. b)
- 1.6.24. a) și e)
- 1.6.25. c)
- 1.6.26. c)
- 1.6.27. c)
- 1.6.28. a), c) și d)
- 1.6.29. e)
- 1.6.30. c) și e)
- 1.6.31. b)

- 1.6.32. e)
- 1.6.33. a) și e)
- 1.6.34. b) și c)
- 1.6.35. c)
- 1.6.36. c)
- 1.6.37. c) și e)
- 1.6.38. c)
- 1.6.39. a) și c)
- 1.6.40. a), d) și e)
- 1.6.41. b)
- 1.6.42. d)
- 1.6.43. d)
- 1.6.44. b)
- 1.6.45. b)
- 1.6.46. b) și c)
- 1.6.47. a)
- 1.6.48. b)
- 1.6.49. d)
- 1.6.50. b) și e)
- 1.6.51. c)
- 1.6.52. d)
- 1.6.53. b), c) și d)
- 1.6.54. e)
- 1.6.55. b)
- 1.6.56. b) și c)
- 1.6.57. a), c) și e)
- 1.6.58. b) și d)
- 1.6.59. b)
- 1.6.60. b)
- 1.6.61. c)
- 1.6.62. a), b) și e)

# CAPITOLUL 2

## Tablouri unidimensionale și bidimensionale

### 2.1. Tablouri în limbajul C++

Numim *tablou* o colecție de date de același tip, în care elementele sunt ordonate, iar accesul la fiecare element are loc prin indice.

În funcție de numărul indicilor avem mai multe tipuri de tablouri:

1. Tablouri unidimensionale (cu un singur indice)
2. Tablouri bidimensionale (cu doi indici)
3. Tablouri multidimensionale (cu mai mulți indici)

#### 2.1.1. Tablouri unidimensionale

Tablourile unidimensionale funcționează ca un vector și se pot declara astfel:



**tip nume\_tablou[dimensiune\_maximă];**

Se observă că este obligatorie folosirea parantezelor drepte care să încadreze dimensiunea maximă pe care o alege utilizatorul pentru acel tablou unidimensional.

*Exemplu: Prezentăm în continuare câteva declarări de tablouri unidimensionale:*



Exemplu:

`int a[25];` // declararea unui tablou unidimensional cu maxim 25 de elemente, fiecare de tip întreg;

`float x[30];` // declararea unui tablou unidimensional cu maxim 30 de elemente, fiecare de tip real simplă precizie;

`char s[40];` // declararea unui tablou unidimensional cu maxim 40 de elemente, fiecare de tip caracter:

Compilatorul C++ alocă un spațiu de memorie egal cu numărul maxim de elemente ale tabloului, rezervând bineînțeles octeți în funcție de tipul de bază al fiecărui tablou. Accesul la fiecare element al tabloului se face prin numele acestuia urmat între paranteze, de indicele său (adică poziția pe care o ocupă în tablou). În limbajul C++, indicii tablourilor încep numărătoarea de la valoarea 0.

*Exemplu: Prezentăm în continuare câteva modalități de acces la elementele ce pot fi memorate în tablourile unidimensionale declarate anterior:*



Exemplu:

**a[0]** – reprezintă elementul aflat pe prima poziție în tablou

**a[24]** - reprezintă elementul aflat pe ultima poziție în tablou

**x[i]** - reprezintă elementul aflat pe poziția i în tablou, unde i poate avea valori între 0 și 29.

Inițializarea elementelor unui tablou se poate face total sau parțial în declararea lor:

```
int b[5]={1,2,3,4,5};
```

Astfel, elementul b[0] are valoarea 1, elementul b[1] are valoarea 2, elementul b[2] are valoarea 3, elementul b[3] are valoarea 4, elementul b[4] are valoarea 5.

În continuare prezentăm câteva programe cu tablouri unidimensionale:

**Problema 1:** Se consideră  $n$  numere reale. Se cere să se determine valoarea minimă și valoarea maximă.



```
#include<stdio.h>
int main(void)
{
    int i,n;
    float x[50],min,max;
    printf("Dati numarul de elemente ale tabloului ");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        printf("x[%d]= ",i);
        scanf("%f",&x[i]);
    }
    min=x[1];max=x[1];
    for(i=2;i<=n;i++)
        if(min>x[i]) min=x[i];
        else if(max<x[i]) max=x[i];
    printf("\nMinimul este %.2f",min);
    printf("\nMaximul este %.2f",max);
}
```

**Problema 2:** Se consideră  $n$  numere întregi. Se cere să se verifice dacă ele sunt sau nu în ordine crescătoare, afișând câte un mesaj corespunzător.



```
#include<stdio.h>
int main(void)
{
    int i,n,verif=1;
    int x[50];
    printf("Dati numarul de elemente ale tabloului X ");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        printf("x[%d]= ",i);
        scanf("%d",&x[i]);
    }
    for(i=1;i<=n-1;i++)
        if( x[i]>x[i+1] ) verif=0;
    if(verif==1) printf("\nNumerele din tablou sunt in ordine
                           CRESCATOARE");
    else printf("\nNumerele din tablou NU sunt in ordine
                           CRESCATOARE");
}
```

## 2.1.2. Tablouri bidimensionale

Tablourile bidimensionale funcționează ca o matrice și se pot declara astfel:



**tip nume\_tablou[dimensiune\_linie][dimensiune\_coloană];**

La fel ca și în cazul tablourilor unidimensionale, și în cazul tablourilor bidimensionale, la declarare, se trece dimensiunea maximă a liniilor (dimensiune\_linie) și dimensiunea maximă a coloanelor(dimensiune\_coloana).

*Exemplu: Prezentăm în continuare câteva declarări de tablouri bidimensionale:*



Exemplu:

```
int a[10][10]; // declararea unui tablou bidimensional cu  
// maxim 100 de elemente (10*10), fiecare de tip întreg;  
float x[5][5]; // declararea unui tablou bidimensional cu  
// maxim 25 de elemente(5*5), fiecare de tip real simplă precizie;  
char s[20][10]; // declararea unui tablou bidimensional cu  
// maxim 200 de elemente(20*10), fiecare de tip caracter;
```

Compilatorul C++ alocă un spațiu de memorie egal cu numărul de liniînmulțit cu numărul de coloane ale tabloului, rezervând bineînteleș octeți în funcție de tipul de bază al fiecărui tablou. Accesul la fiecare element al tabloului se face prin numele acestuia urmat între paranteze, de indicele liniei și indicele coloanei (adică poziția pe care o ocupă în tablou).

*Exemplu: Prezentăm în continuare câteva modalități de acces la elementele ce pot fi memorate în tablourile bidimensionale declarate anterior:*



Exemplu:

**a[0][0]** – reprezintă elementul aflat pe linia 0 coloana 0  
**a[9][9]** - reprezintă elementul aflat pe linia 9 coloana 9  
**x[i][j]** - reprezintă elementul aflat pe linia i, coloana j în matrice, unde i și j pot avea valori între 0 și 4.

Initializarea elementelor unui tablou se poate face total sau parțial la declararea lor:

```
int b[2][3]={ {1, 2, 3}, {4, 5, 6} };
```

Astfel, elementul  $b[0][0]$  are valoarea 1, elementul  $b[0][1]$  are valoarea 2, elementul  $b[0][2]$  are valoarea 3, elementul  $b[1][0]$  are valoarea 4, elementul  $b[1][1]$  are valoarea 5, elementul  $b[1][2]$  are valoarea 6.

În continuare prezentăm câteva programe cu tablouri bidimensionale:

**Problema 1:** Se consideră două tablouri bidimensionale (matrici) A și B cu  $n \times m$ , respectiv  $m \times p$  numere întregi. Se cere să se calculeze matricea produs :  $C = A * B$ .



```
#include<stdio.h>
int main(void)
{
    int a[10][10],b[10][10],c[10][10];
    int n,m,i,j,k,p;
    printf("Dati dimensiunile matricei A \n");
    printf("Dati numarul de linii n = ");
    scanf("%d",&n);
    printf("Dati numarul de coloane m = ");
    scanf("%d",&m);
    for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)
    {
        printf("a[%d,%d] = ",i,j);
        scanf("%d",&a[i][j]);
    }
    printf("Elementele matricei A sunt : \n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=m;j++) printf("%d ",a[i][j]);
        printf("\n");
    }
}
```

```

printf("Dati dimensiunile matricei B \n");
printf("Dati numarul de linii m = ");scanf("%d",&m);
printf("Dati numarul de coloane p = ");scanf("%d",&p);
for(i=1;i<=m;i++)
{
    for(j=1;j<=p;j++)
    {
        printf("b[%d,%d] = ",i,j);
        scanf("%d",&b[i][j]);
    }
}
printf("Elementele matricei B sunt : \n");
for(i=1;i<=m;i++)
{
    for(j=1;j<=p;j++) printf("%d ",b[i][j]);
    printf("\n");
}
for(i=1;i<=n;i++)
{
    for(j=1;j<=p;j++) c[i][j]=0;
    for(k=1;k<=m;k++)
        c[i][j]=c[i][j] + a[i][k] * b[k][j];
}
printf("Elementele matricei produs sunt : \n");
for(i=1;i<=n;i++)
{
    for(j=1;j<=p;j++) printf("%d ",c[i][j]);
    printf("\n");
}
}

```

*Problema 2: Se consideră o matrice A cu  $n \times m$  numere întregi. Se cere să se obțină transpusa sa.*



```
#include<stdio.h>
int main(void)
{
    int a[10][10],b[10][10];
    int n,m,i,j,min,max;
    printf("Dati dimensiunile matricei A \n");
    printf("Dati numarul de linii n = ");
    scanf("%d",&n);
    printf("Dati numarul de coloane m = ");
    scanf("%d",&m);
    for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)
    {
        printf("a[%d,%d] = ",i,j);
        scanf("%d",&a[i][j]);
    }
    printf("Elementele matricei A sunt : \n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=m;j++) printf("%d ",a[i][j]);
        printf("\n");
    }
    for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)
            b[j][i]=a[i][j];
    printf("Elementele matricei transpuse sunt \n");
    for(i=1;i<=m;i++)
    {
        for(j=1;j<=n;j++) printf("%d ",b[i][j]);
        printf("\n");
    }
}
```

### 2.1.3. Tablouri multidimensionale

Tablourile multidimensionale se pot declara astfel:



```
tip nume_tablou[dimensiune1][dimensiune2]...
[dimensiunen];
```

Compilatorul C++ alocă un spațiu de memorie determinat prin înmulțirea numărului de elemente ale fiecarei dimensiuni în parte (dim1, dim2, dim3, . . . , dimn) cu numărul de octeți ai tipului de date al tabloului respectiv.

*Exemplu: Prezentăm în continuare câteva declarări de tablouri bidimensionale:*



**Exemplu:**

```
int a[10][10][10]; // declararea unui tablou tridimensional cu maxim 1000 de elemente (10*10*10), fiecare de tip întreg;  
float x[5][5][5]; // declararea unui tablou tridimensional cu maxim 125 de elemente(5*5*5), fiecare de tip real simplă precizie;  
char s[20][10][30]; // declararea unui tablou tridimensional cu maxim 6000 de elemente(20*10*30), fiecare de tip caracter;
```

*Exemplu: Prezentăm în continuare, un program care afișează numărul de octeți alocați pentru un tablou unidimensional, un tablou bidimensional și un tablou multidimensional ( cu trei dimensiuni ), utilizând operatorul sizeof:*



```
#include <stdio.h>
int main(void)
{
    int a[20];
    float x[3][4];
    char t[20][30][40];
    printf("Numărul de octeți pentru tabloul unidimensional int a[20] este %d\n", sizeof(a));
    printf("Numărul de octeți pentru tabloul bidimensional float x[3][4] este %d\n", sizeof(x));
    printf("Numărul de octeți pentru tabloul multidimensional char t[20][30][40] este %d\n", sizeof(t));
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

*Numărul de octeți pentru tabloul unidimensional int a[20] este 40*

*Numărul de octeți pentru tabloul bidimensional float x[3][4] este 48*

*Numărul de octeți pentru tabloul multidimensional char t[20][30][40] este 48000*

*În continuare prezentăm un program cu un tablou multidimensionale:*

**Problema:** Se consideră un cub împărțit în  $n^3$  cuburi mai mici (toate de aceleași dimensiuni),  $n$  număr natural nenul dat. Fiecarui cub i se asociază un număr natural. Să se determine suma numerelor asociate cuburilor de pe cele 4 diagonale ale cubului mare.



```
#include<stdio.h>
int main(void)
{
    int a[10][10][10];
    int n,i,j,k,suma=0;
    printf("Dati dimensiunile matricei A \n");
    printf("Dati numarul n = ");scanf("%d",&n);
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            for(k=1;k<=n;k++)
            {
                printf("a[%d,%d,%d] = ",i,j,k);
                scanf("%d",&a[i][j][k]);
            }
    for(i=1;i<=n;i++)
        suma+=a[i][i][i]+a[i][n-i+1][n-i+1]+a[i][n-i+1][i]+a[i][i][n-i+1];
    printf("suma=%d",suma);
}
```

## 2.2. Probleme rezolvate

### 2.2.1. Enunțuri:

1. Fie tabloul unidimensional  $x$  cu  $n$  elemente numere reale și numerele întregi  $a$  și  $b$ . Să se calculeze media aritmetică a elementelor din tablou cuprinse între valorile  $a$  și  $b$ .
2. Se dă un tablou unidimensional cu  $n$  componente numere întregi și un număr întreg  $A$ . Să se numere câte elemente sunt mai mari decât  $A$  și să se construiască un vector cu aceste elemente.

3. Se dă un tablou unidimensional  $X$  cu  $n$  numere reale și se cere să se modifice astfel încât să se intercaleze între oricare două elemente consecutive, media lor aritmetică.
4. Se consideră un tablou unidimensional  $x$  cu  $n$  numere întregi. Se cere să se calculeze cel mai mare divizor comun al elementelor tabloului.
5. Se consideră un tablou unidimensional cu  $n$  numere reale și se cere să se afișeze cel mai mare și cel mai mic element din tablou.
6. Se citesc două tablouri unidimensionale cu componente numere naturale. Fiecare tablou are elementele sortate crescător. Se cere să se construiască un al treilea tablou care conține elementele celor două tablouri date, în ordine crescătoare. (*Problema interclasării*)
7. Să se scrie un program care calculează produsul a două matrici (tablouri bidimensionale)  $A_{n*m}$ ,  $B_{m*p}$ .
8. Să se scrie un program care calculează minimul și maximul dintr-o matrice cu  $n$  linii și  $m$  coloane ( $A_{n*m}$ ) ( $1 \leq n, m \leq 30$ ).
9. Să se scrie un program care calculează transpusa unei matrici  $A_{n*m}$  ( $1 \leq n, m \leq 30$ ).
10. Se consideră o matrice  $A_{n*n}$ . Să se calculeze:
  - a. suma elementelor de pe diagonala principală
  - b. produsul elementelor de pe diagonala secundară
  - c. minimele din elementele aflate deasupra, respectiv sub diagonala principală
  - d. maximele din elementele aflate deasupra, respectiv sub diagonala secundară
11. Se consideră o matrice  $A_{n*m}$  ( $1 \leq n, m \leq 30$ ) cu elemente numere întregi. Să se determine linia (liniile) din matrice care conține (conțin) cele mai multe elemente nenule.

## 2.2.2. Soluții propuse:

### Problema 1:

Fie tabloul unidimensional x cu n elemente numere reale și numerele întregi a și b. Să se calculeze media aritmetică a elementelor din tablou cuprinse între valorile a și b.



```
#include<iostream.h>
int main(void)
{
    int a,b,i,n,suma=0;
    float x[50],media;
    cout<<"Dati numarul de elemente ale tabloului "; cin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"x["<<i<<"]= ";
        cin>>x[i];
    }
    cout<<"Dati numarul a = ";cin>>a;
    cout<<"Dati numarul b = ";cin>>b;
    for(i=1;i<=n;i++)
        if( (a<=x[i]) && (x[i]<=b) ) suma+=x[i];
    media=suma/n;
    cout<<"Media aritmetica a elementelor din tablou, aflate
    intre "<<a<<" si "<<b<<" este "media;
}
```

### Problema 2:

Se dă un tablou unidimensional cu n componente numere întregi și un număr întreg A. Să se numere câte elemente sunt mai mari decât A și să se construiască un vector cu aceste elemente.



```
#include<iostream.h>
int main(void)
{
    int i,n,j,a;
    float x[50],y[50];
    cout<<"Dati numarul de elemente ale tabloului "; cin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"x["<<i<<"]= ";
        cin>>x[i];
    }
    cout<<"Dati numarul a = ";cin>>a;
    j=0;
    for(i=1;i<=n;i++)
        if (a<=x[i]) { j++; y[j]=x[i]; }
    cout<<"Sunt "<<j<<" numere mai mari decat
"<<a<<endl;
    cout<<"Elementele sunt: "<<endl;
    for(i=1;i<=j;i++) cout<<y[i]<<" ";
}
```

### Problema 3:

Se dă un tablou unidimensional X cu n numere reale și se cere să se modifice astfel încât să se intercaleze între oricare două elemente consecutive, media lor aritmetică.



```
#include<iostream.h>
int main(void)
{
    int i,n,j;
    float x[50],y[50];
    cout<<"Dati numarul de elemente ale tabloului ";cin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"x["<<i<<"]= ";
        cin>>x[i];
    }
    i=1; j=1;
    while(i<=n)
    {
        y[j]=x[i];
        y[j+1]=(x[i]+x[i+1])/2;
        i=i+1; j=j+2;
    }
    cout<<"Elementele sunt: "<<endl;
    for(i=1;i<=j;i++) cout<<y[i]<<" ";
}
```

#### Problema 4:

Se consideră un tablou unidimensional x cu n numere întregi. Se cere să se calculeze cel mai mare divizor comun al lor.



```
#include<iostream.h>
int main(void)
{
    int i,n,cmmdc,r,d;
    int x[50];
    cout<<"Dati numarul de elemente ale tabloului ";
    cin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"x["<<i<<"]= ";
        cin>>x[i];
    }
    r=x[1]/x[2];
    while(r!=0)
    {
        x[1]=x[2];
        x[2]=r;
        r=x[1]/x[2];
    }
    d=x[1];
    for(i=3;i<=n;i++)
    {
        r=d/x[i];
        while(r!=0)
        {
            d=x[i];
            x[i]=r;
            r=d/x[i];
        }
        d=x[i];
    }
    cmmdc=d;
    cout<<endl<<" C.m.m.d.c este "<<cmmdc;
}
```

### Problema 5:

Se consideră un tablou unidimensional cu n numere reale și se cere să se afișeze cel mai mare și cel mai mic element din tablou. (minimul și maximul)



```
#include<iostream.h>
int main(void)
{
    int i,n;
    float x[50],min,max;
    cout<<"Dati numarul de elemente ale tabloului ";
    cin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"x["<<i<<"]= ";
        cin>>x[i];
    }

    min=x[1];max=x[1];
    for(i=1;i<=n;i++)
        if(min>x[i]) min=x[i];
        else if(max<x[i]) max=x[i];
    cout<<endl<<"Minimul este "<<min<<endl;
    cout<<"Maximul este "<<max;
}
```

### Problema 6:

Se citesc două tablouri unidimensionale cu componente numere naturale. Fiecare tablou are elementele sortate crescător. Se cere să se construiască un al treilea tablou care conține elementele celor două în ordine crescătoare. (*Problema interclasării*)



```
#include<iostream.h>
int main(void)
{
    int i,n,j,m,k;
    float x[50],y[50],z[100];
    cout<<"Dati numarul de elemente ale tabloului X ";
    cin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"x["<<i<<"]= ";
        cin>>x[i];
    }
```

```

cout<<"Dati numarul de elemente ale tabloului Y ";
cin>>m;
for(j=1;i<=m;j++)
{
    cout<<"y["<<j<<"]= ";
    cin>>y[j];
}
i=1;j=1;k=0;
while( (i<=n) && (j<=m) )
{
    if(x[i]<y[j]) {k++;z[k]=x[i];i++;}
    else {k++;z[k]=y[j];j++;}
if(i<=n) for(j=i;j<=n;j++) {k++;z[k]=x[j];}
else for(i=j;i<=m;i++) {k++;z[k]=y[i];}
cout<<"\nVectorul Z cu elementele interclasate este ";
for(i=1;i<=k;i++) cout<<" " <<z[i];
}

```

### Problema 7:

Să se scrie un program care calculează produsul a două matrici (tablouri bidimensionale)  $A_{n*m}$ ,  $B_{m*p}$ .



```

#include <iostream.h>
int main(void)
{
    int n,m,p,i,j,k,a[10][10],b[10][10],c[10][10];
    cout<<"Dati dimensiunile matricei "<<endl;
    cout<<"Dati numarul de linii n = ";    cin>>n;
    cout<<"Dati numarul de coloane m = ";  cin>>m;
    for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)
    {
        cout<<"a["<<i<<","<<j<<"]= ";
        cin>>a[i][j];
    }
    cout<<"Elementele matricei A sunt: "<<endl;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=m;j++)
            cout<<a[i][j]<<" ";
        cout<<endl;
    }
}

```

```

cout<<"Dati numarul de linii m = "; cin>>m;
cout<<"Dati numarul de coloane p = "; cin>>p;
for(i=1;i<=m;i++)
{
    for(j=1;j<=p;j++)
    {
        cout<<"b["<<i<<","<<j<<"] = ";
        cin>>b[i][j];
    }
    cout<<"Elementele matricei B sunt: "<<endl;
}
for(i=1;i<=m;i++)
{
    for(j=1;j<=p;j++)
        cout<<b[i][j]<<" ";
    cout<<endl;
}

for(i=1;i<=n;i++)
{
    for(j=1;j<=p;j++)
        for(k=1;k<=m;k++)
            c[i][j]+=a[i][k]*b[k][j];
    cout<<"Elementele matricei produs "<<endl;
}
for(i=1;i<=n;i++)
{
    for(j=1;j<=p;j++)
        cout<<c[i][j]<<" ";
    cout<<endl;
}
}

```

### Problema 8:

Să se scrie un program care calculează minimul și maximul dintr-o matrice cu n linii și m coloane ( $A_{n*m}$ ) ( $1 \leq n, m \leq 30$ ).



```

#include <iostream.h>
int main(void)
{
    int n, m, min, i, j, max, a[30][30];
    cout<<"Dati dimensiunile matricei "<<endl;
    cout<<"Dati numarul de linii n = "; cin>>n;
    cout<<"Dati numarul de coloane m = "; cin>>m;

```

```

for(i=1;i<=n;i++)
    for(j=1;j<=m;j++)
    {
        cout<<"a["<<i<<","<<j<<"]= ";
        cin>>a[i][j];
    }
cout<<"Elementele matricei A sunt: "<<endl;
for(i=1;i<=n;i++)
{
    for(j=1;j<=m;j++)
        cout<<a[i][j]<<" ";
    cout<<endl;
}
min = a[1][1];  max = a[1][1];
for(j=1;i<=m;i++)
{
    if( a[1][j] < min )      min = a[1][j];
    if( a[1][j] > max )      max = a[1][j];
}
for(i=2;i<=n;i++)
    for(j=1;j<=m;j++)
    {
        if( a[i][j] < min )      min = a[i][j];
        if( a[i][j] > max )      max = a[i][j];
    }
cout<<"Elementul minim din matrice este
"<<min<<endl;
cout<<"Elementul maxim din matrice este
"<<max<<endl;
}

```

### Problema 9:

Să se scrie un program care calculează transpusa unei matrici  $A_{n*m}$  ( $1 \leq n, m \leq 30$ ).



```

#include <iostream.h>
int main(void)
{
    int n,m,i,j,a[30][30],b[30][30];
    cout<<"Dati dimensiunile matricei "<<endl;
    cout<<"Dati numarul de linii  n = ";      cin>>n;
    cout<<"Dati numarul de coloane m = ";  cin>>m;
}

```

```

for(i=1;i<=n;i++)
    for(j=1;j<=m;j++)
    {
        cout<<"a["<<i<<","<<j<<"]= ";
        cin>>a[i][j];
    }
    cout<<"Elementele matricei A sunt: "<<endl;
for(i=1;i<=n;i++)
{
    for(j=1;j<=m;j++)
        cout<<a[i][j]<<" ";
    cout<<endl;
}
for(i=1;i<=n;i++)
    for(j=1;i<=m;j++)
        b[j][i]=a[i][j];
cout<<"Matricea transpusa este "<<endl;
for(i=1;i<=n;i++)
{
    for(j=1;j<=m;j++)
        cout<<b[i][j]<<" ";
    cout<<endl;
}
}

```

### Problema 10:

Se consideră o matrice  $A_{n \times n}$ . Să se calculeze:

- suma elementelor de pe diagonala principală
- produsul elementelor de pe diagonala secundară
- minimele din elementele aflate deasupra, respectiv sub diagonala principală
- maximele din elementele aflate deasupra, respectiv sub diagonala secundară



```

#include <iostream.h>
int main(void)
{
    int n,m,i,j,a[30][30],min1,min2,max1,max2,suma,produs;
    cout<<"Dati dimensiunile matricei "<<endl;
    cout<<"Dati numarul de linii si de coloane n = ";
    cin>>n;

```

```

for(i=1;i<=n;i++)
    for(j=1;j<=n;j++)
        { cout<<"a["<<i<<","<<j<<"]= ";
          cin>>a[i][j]; }
    cout<<"Elementele matricei A sunt: "<<endl;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
            cout<<a[i][j]<<" ";
        cout<<endl; }
// ===== suma elementelor de pe diagonala principala =====
suma=0;
for(i=1;i<=n;i++)
    for(j=1;i<=n;j++)
        if(i==j) suma+=a[i][j];
cout<<"Suma elementelor de pe diagonala principala este
"<<suma<<endl;
// ===== produsul elementelor de pe diagonala secundara =====
produs=1;
for(i=1;i<=n;i++)
    for(j=1;i<=n;j++) if(i+j==n+1) produs*=a[i][j];
cout<<"Produsul elementelor de pe diagonala secundara este
"<<produs<<endl;
// ===== minimul elementelor de deasupra diagonalei principale ==
min1=32768;
for(i=1;i<=n;i++)
    for(j=1;j<=m;j++) if(i<j)
        if(min1>a[i][j]) min1=a[i][j];
cout<<"Minimul de deasupra diag. principale este "<<min1;
// ===== minimul elementelor de sub diagonala principale ====
min2=32768;
for(i=1;i<=n;i++)
    for(j=1;j<=m;j++) if(i>j)
        if(min2>a[i][j]) min2=a[i][j];
cout<<"Minimul de sub diag. principala este "<<min2;
// == maximul elementelor de deasupra diagonalei principale ===
max1=-32768;
for(i=1;i<=n;i++)
    for(j=1;j<=m;j++)
        if(max1<a[i][j]) max1=a[i][j];
cout<<"Maximul de deasupra diag. principale este "<<max1;
// == maximul elementelor de sub diagonala principala ====
max2=-32768;
for(i=1;i<=n;i++)
    for(j=1;j<=m;j++)
        if(max2<a[i][j]) max2=a[i][j];
cout<<"Minimul de sub diag. principala este "<<max2;
}

```

### Problema 11:

Se consideră o matrice  $A_{n*m}$  ( $1 \leq n, m \leq 30$ ) cu elemente numere întregi. Să se determine linia (liniile) din matrice care conține (conțin) cele mai multe elemente nenule.



```
#include<iostream.h>
int main(void)
{
    int a[30][30], n, m, i, j, max, nr;
    cout<<"Dati numarul de linii n = "; cin>>n;
    cout<<"Dati numarul de coloane m = "; cin>>m;
    for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)
    {
        cout<<"a["<<i<<","<<j<<"] = ";
        cin>>a[i][j];
    }

    cout<<endl<<"Matricea A are elementele:"<<endl;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=m;j++) cout<<a[i][j]<<" ";
        cout<<endl;
    }

    max=0;
    for(i=1;i<=n;i++)
    {
        nr=0;
        for(j=1;j<=m;j++)
            if(a[i][j]!=0) nr++;
        if(max<nr) max=nr;
    }

    for(i=1;i<=n;i++)
    {
        nr=0;
        for(j=1;j<=m;j++)
            if(a[i][j]!=0) nr++;
        if(max==nr) cout<<"Linia "<<i<<" are
"<<max<<" elemente nenule"<<endl;
    }
}
```

## 2.3. Probleme propuse spre rezolvare

1. Scrieți un program care afișează în ordine inversă componentele unui tablou **a** format din 10 elemente (numere întregi).
2. Scrieți secvența de instrucțiuni necesară pentru înmulțirea  $\mathbf{a}^T * \mathbf{b}$ , unde **a** și **b** sunt doi vectori de aceeași dimensiune.
3. Scrieți un program care normalizează un vector dat, **v**, de dimensiune dată, **n**, adică împarte fiecare componentă a vectorului prin valoarea absolută maximă depistată prin explorarea valorilor absolute ale tuturor componentelor.
4. De la tastatură se introduc valorile componentelor a două matrice numerice, **a** și **b**. Să se calculeze și să se afișeze elementele matricei **s** ce reprezintă suma matricelor **a** și **b**.
5. Să se afișeze un vector **v** având **n** componente, scriind câte **p** componente pe un rând.
6. Fiind cunoscută matricea numerică pătrată **m** de dimensiune **n\*n** precizată, scrieți un program care stabilește dacă:
  - **m** este simetrică față de diagonala principală;
  - **m** este inferior triunghiulară, adică toate elementele situate deasupra diagonalei principale sunt nule.
7. Pentru o matrice numerică formată din **n** linii și **m** coloane ( $1 < n, m < 10$ ) să se scrie un program care afișează liniile conținând **k** elemente nule ( $0 < k \leq m$ ). Se va afișa un mesaj în situația în care nici o linie nu conține exact **k** elemente nule.

8. Se dau două matrice de numere întregi: **a** cu  $ma$  linii și  $ma$  coloane și **b** cu  $mb$  linii și  $mb$  coloane, astfel încât  $ma \geq mb$  și  $na \geq nb$ . Să se decidă dacă **b** este o submatrice a lui **a**, adică dacă există  $k, l$  astfel încât :  $a_{k+i-1, l+j-1} = b_{i, j}$  cu  $i=1, \dots, mb$  și  $j=1, \dots, nb$ . În caz afirmativ se vor tipări **k** și **l**.
9. Să se scrie un program care, primind un sir **x** de numere întregi cu  $n$  elemente, neordonate, și o valoare întreagă **v**, decide dacă **v** se află sau nu în sir. În caz afirmativ, tipărește toate pozițiile pe care se află valoarea **v**. În caz contrar, tipărește un mesaj corespunzător.
10. Într-un sir de numere **s** cu  $n$  elemente, să se determine elementele cu o singură apariție. Elementele sirului se citesc de la tastatură, precedate de numărul lor,  $n$  ( $n \leq 100$ ).
11. Dintr-o matrice numerică **a** cu  $l$  linii și  $c$  coloane să se afișeze liniile care reprezintă siruri ordonate crescător și coloanele care reprezintă siruri ordonate descrescător.
12. Se dă un sir de numere cu  $n$  elemente. Să se determine valorile maximă și respectiv minimă existente în sir, precum și pozițiile în care ele apar. Se are în vedere și situația când valorile apar în mai multe poziții.
13. De la tastatură se citește un sir de numere **x** cu  $n$  componente, ordonat strict crescător și o valoare **y**. Sa se insereze această valoare în sirul **x** astfel încât el să rămână ordonat strict crescător.
14. Dându-se o valoare **x** și un tablou de numere **a** cu  $n$  elemente, să se separe acest tablou în două partiții astfel încât elementele din prima partitură să fie mai mici sau egale cu **x**, iar cele din a doua partitură să fie mai mari decât **x**.
15. Se dau două siruri de numere întregi: **x** cu  $nx$  elemente și **y** cu  $ny$  elemente,  $nx > ny$ . Să se decidă dacă **y** este un subșir al lui **x**, adică dacă există un număr **k** astfel încât:

$$x_k = y_1$$

$$x_{k+1} = y_2$$

...

$$x_{k+ny-1} = y_n$$

In caz afirmativ se va tipări valoarea lui k.

16. O matrice de numere, notată **a**, are p linii și q coloane. Să se creeze o nouă matrice **b** din matricea **a**, exceptând liniile și coloanele la intersecția cărora se află elemente nule. Se vor utiliza doi vectori de numere în care se vor marca liniile, respectiv coloanele care urmează să nu mai apară în **b**.
17. Se dau două șiruri **x** și **y** ordonate strict crescător, având **m** și respectiv **n** elemente. Să se construiască un nou șir **z** ordonat strict crescător care să conțină elementele șirurilor **x** și **y** ("interclasare de șiruri").
18. De la tastatură se citește un șir de numere, notat **x**, cu **n** elemente ordonate strict crescător. Să se insereze în acest șir un număr neprecizat de valori **alfa** citite de la tastatură, astfel încât șirul să rămână ordonat strict crescător. Procesul de inserare începează în momentul în care **alfa < x<sub>1</sub>**, **x<sub>1</sub>** fiind primul element al șirului **x**.
19. Se dă o matrice, notată **a**. Să se scrie un program care afișează elementele maximale de pe coloane, ordonează descrescător liniile matricei după primul element din fiecare linie a acesteia și apoi afișează matricea astfel ordonată.
20. Se citește un tablou unidimensional cu n ( $1 \leq n \leq 100$ ) componente numere naturale. Se cere să se construiască și să se afișeze un nou vector cu componentele *pătrate perfecte* ale vectorului inițial.
21. Se dă un tablou unidimensional cu n ( $1 \leq n \leq 100$ ) componente numere naturale. Să se calculeze suma componentelor divizibile cu a, pentru un număr a citit de la tastatură.

**22.** Se dă un tablou unidimensional cu  $n$  ( $1 \leq n \leq 100$ ) componente numere naturale. Să se verifice dacă componentele sunt în ordine crescătoare de la stânga la dreapta.

**23.** Să se rearanjeze elementele unui tablou unidimensional de numere întregi, astfel încât numerele pare să apară înaintea numerelor impare. În cadrul subsecvenței de numere pare, respectiv impare, elementele trebuie să apară în ordinea în care erau în vectorul inițial.

**24.** Fiind date două tablouri unidimensionale  $a$  și  $b$  cu,  $n$  respectiv  $m$  elemente numere întregi, să se scrie un program C++ care să verifice dacă au aceleași elemente distincte (toate elementele unui tablou se găsesc printre elementele celuilalt tablou și invers).

*Exemplu:* Pentru  $n = 5$  și  $a = \{2, 3, 2, 4, 1\}$

$$m = 6 \text{ și } b = \{2, 4, 2, 4, 1, 3\}$$

au aceleași elemente distincte :  $(1, 2, 3, 4)$ .

**25.** Se consideră o matrice  $A_{n*m}$  ( $1 \leq n, m \leq 30$ ) având componente numere întregi. Se cere să se calculeze suma componentelor de pe marginea matricei.

**26.** Se consideră o matrice  $A_{n*m}$  ( $1 \leq n, m \leq 30$ ) având componente numere întregi. Să se determine maximul fiecărei coloane și minimul fiecărei linii.

**27.** Să se construiască o matrice pătratică ( $n=m$ ) de dimensiune  $n^2$  ( $1 \leq n \leq 30$ ) cu primele numere pare începând cu 2.

**28.** Fiind dat un tablou bidimensional  $A(n,m)$  de numere întregi, să se calculeze și să afișeze suma elementelor aflate pe marginea (rama) tabloului.

*Exemplu:* Pentru  $n = 3$  și  $m = 4$  și  $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 10 & 11 & 12 & 5 \\ 9 & 8 & 7 & 6 \end{pmatrix}$

se va afișa suma  $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 47$ .

**29.** Scrieți un program care să realizeze inversarea unui vector:

- a) în același vector și fără a utiliza un vector suplimentar;
- b) într-un alt vector

*Exemplu:* Dacă vectorul inițial este  $x = \{1, 2, 3, 4\}$ , atunci cel inversat va fi  $\{4, 3, 2, 1\}$

**30.** Fiind dat un vector  $v$  cu  $n$  elemente numere întregi, să se construiască alți doi vectori : primul va conține numai elementele pare, iar al doilea numai elementele impare ale vectorului inițial.

*Exemplu :* Dacă vectorul inițial este  $v = \{1, 64, 2, 5, 23, 9, 6, 11\}$ , se vor obține vectorii  $x = \{64, 2, 6\}$  și  $y = \{1, 5, 23, 9, 11\}$

**31.** Să se verifice dacă un vector conține elemente în ordinea : negativ, negativ, pozitiv, negativ, negativ, pozitiv, etc.

*Exemplu :* Vectorul  $\{-1, -2, 3, -5, -6, 2, -9\}$  îndeplinește cerința problemei ; vectorul  $\{-1, 2, -3, -4\}$  nu îndeplinește cerința problemei.

**32.** Să se calculeze cel mai mare divizor comun a  $n$  numere - cmmdc ( $x_1, x_2, \dots, x_n$ ).

*Exemplu :* cmmdc(2940,882,70,182) = 14.

**33.** Să se afișeze câte elemente dintr-un vector de numere întregi sunt prime cu un număr dat.

*Exemplu :* Fie vectorul  $v = \{12, 15, 254, 525, 56, 125, 500, 63, 48, 912\}$  și numărul 4, atunci se obține următorul rezultat : 4 numere îndeplinesc cerința problemei și acestea sunt  $\{15, 525, 125, 63\}$ .

**34.** Scrieți un program care elimină toate elementele nule dintr-un vector de numere întregi.

*Exemplu :* Dacă vectorul inițial este  $x = \{10, 0, 0, 3, 0, 5, 0, 0, 8\}$  se va obține  $x = \{10, 3, 5, 8\}$

**35.** Se consideră un sir de n numere reale. Să se scrie un program care elimină din sir valorile ce se află în afara intervalului [a,b], cu a și b citiți de la tastatură.

*Exemplu :* Dacă sirul inițial este  $x = \{2,8,4,6,9,10,3,5,2,10\}$  și  $a=3$ ,  $b=7$ , atunci vectorul rezultat va fi  $\{4,6,3,5\}$ .

**36.** Dându-se n numere întregi să se decidă dacă există un *număr majoritar* în această secvență. Un număr este *majoritar* dacă numărul său de apariții în vector este mai mare decât  $n/2$ .

*Exemplu :* În vectorul  $x=\{2,4,3,2,2,2,6\}$  numărul 2 este element majoritar.

**37.** Se citesc n numere naturale. Aceste numere se împart în grupe astfel încât în cadrul fiecarei grupe toate numerele au același număr de cifre 1 în reprezentarea în baza 2. Se cere să se afișeze mediile aritmetice a numerelor din fiecare grupă.

*Exemplu :* Pentru  $n=10$  și vectorul  $\{92,60,47,16,52,45,65,7,8,87\}$  se obțin următoarele medii aritmetice : 12, 65, 29.5, 65.55, 67.

**38.** Să se afișeze toate tripletele de numere crescătoare de pe poziții consecutive din vectorul x de numere reale.

*Exemplu :* Pentru vectorul  $x=\{2,9,41,61,6,24,84,1,21\}$  se vor afișa tripletele  $(2,9,41)$ ,  $(9,41,61)$  și  $(6,24,84)$ .

**39.** Fie un vector x de numere întregi. Să se afișeze toate perechile de numere consecutive din x, cu proprietatea că al doilea reprezintă numărul de apariții ale cifrei 3 în pătratul primului.

*Exemplu :* Pentru vectorul  $x=\{361, 2, 5, 1156, 4, 0\}$ , perechile afișate vor fi  $(361,2)$ ,  $(1156,4)$ ,  $(4,0)$ .

**40.** Să se scrie un program care verifică dacă un vector x cu n componente numere întregi este o permutare a mulțimii  $\{1, 2, 3, \dots, n\}$ .

*Exemplu :* Vectorul  $x=\{2,6,1,4,3,5\}$  reprezintă o permutare a mulțimii  $\{1,2,3,4,5,6\}$ .

**41.** Să se transforme un număr din baza 10 în baza 16, cifrele numărului în baza 16 memorându-se într-un vector.

*Exemplu :* Pentru  $n=11224$  se obține vectorul cu resturile împărțirilor  $(2,11,13,8)$ , deci numărul în baza 16 este  $2BD8$ .

**42.** Să se determine numărul de cifre egale cu cifra c (dată de la tastatură) în cadrul numărului n! cu  $n \leq 500$ .

*Exemplu :* Cifra 2 apare de 3 ori în numărul  $13!$  ( $13! = 6227020800$ ).

**43.** Se dă un vector cu n componente numere întregi. Să se determine diferența maximă dintre două elemente consecutive ale acestui vector.

*Exemplu :* Dacă  $x=\{15,36,94,15,64,32,245,11,260,33,56\}$ , atunci diferența maximă este 249, aflată între valorile (11,260).

**44.** Se dau doi vectori  $\{a_1, a_2, \dots, a_n\}$  și  $\{b_1, b_2, \dots, b_n\}$ . Dacă unul din vectori este permutare circulară a celuilalt, să se insereze în vectorul a după fiecare element par toate elementele mai mari decât el din vectorul inițial.

*Exemplu:* Pentru vectorii  $a = \{5, 2, 8, 1, 3\}$  și  $b = \{1, 3, 5, 2, 8\}$ , se va obține vectorul  $a = \{5, 2, 5, 8, 3, 8, 1, 3\}$

**45.** Fie n un număr natural nenul. Să se construiască vectorul  $x = \{x_1, x_2, \dots, x_n\}$  ce conține primii n termeni consecutivi ai șirului:

$$1, 1, 2, 1, 2, 3, 4, 4, 4, 4, 1, 2, 3, 4, 5, 6, 6, \dots$$

Obținut din șirul numerelor naturale prin înlocuirea fiecărui număr natural prim a cu secvența  $1, 2, \dots, a$  și a fiecărui număr natural neprim b prin scrierea de b ori a numărului b.

**46.** Se consideră trei numere naturale x, y, z, care împărțite la același număr natural n dau resturile 7, 9, respectiv 11.

- determinați cel mai mare număr care îndeplinește condițiile de mai sus;
- determinați cel mai mic număr care îndeplinește condițiile de mai sus;

*Exemplu:* Fie numerele 247, 297 și 347.

- 48 (număr maxim)

b) 12 (număr minim)

Indicație:

a) Se aplică teorema împărțirii cu rest:

$$247 : n \Rightarrow \text{câtul } c_1 \text{ și restul } 7 \quad n * c_1 = 240$$

$$297 : n \Rightarrow \text{câtul } c_2 \text{ și restul } 9 \quad n * c_2 = 288$$

$$347 : n \Rightarrow \text{câtul } c_3 \text{ și restul } 11 \quad n * c_3 = 336$$

$$\Rightarrow \text{număr maxim} = \text{cmmdc}(240, 288, 336)$$

b) Dacă restul maxim este 11, rezultă că numărul minim căutat este  $n > 11$  (împărțitorul mai mare decât restul), deci nu ne rămâne decât să căutăm între 12 și numărul maxim (de la punctul a)) primul număr pentru care  $(x-7)$ ,  $(y-9)$  și  $(z-11)$  se divid la el.

47. Fie A o matrice pătratică, de dimensiune n, care este definită astfel:

$$a[i][j] = \begin{cases} 0, \text{ dacă } i^2 + j^2 \text{ este număr prim} \\ 1, \text{ în caz contrar} \end{cases}$$

Considerând că fiecare linie a matricii este o reprezentare în baza 2 a unui număr natural, să se afișeze cel mai mare număr corespunzător liniilor matricei.

Exemplu: Pentru  $n = 4$ , matricea corespunzătoare va fi:

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

Cel mai mare număr natural cu proprietatea cerută este 10.

48. Se dau valorile m și n numere naturale nenule. Să se formeze matricea A cu m linii și n coloane cu elementele sirului:

1,2,2,1,2,3,4,4,4,4,1,2,3,4,5,6,6,6,6,6,1,2,3,4,5,6,7,8,8,8,8,8,8,1,...

**49.** Fie  $m$  și  $n$  două numere naturale nenule,  $x$  un sir de  $n$  numere reale, iar  $y$  un sir de  $m$  numere reale. Să se formeze matricea  $A$  (având  $m$  linii și  $n$  coloane) cu elemente numere reale, în felul următor:

$$a[i][j] = \begin{cases} \max\{x_i, \dots, x_j, y_i, \dots, y_j\}, & \text{dacă } i \leq j. \\ \min\{x_j, \dots, x_i, y_j, \dots, y_i\}, & \text{în caz contrar.} \end{cases}$$

**50.** Se dau  $m$  și  $n$  două numere naturale nenule. Să se formeze matricea  $A$ , de dimensiuni  $m \times n$ , din elementele sirului:

1, 1, 2, 4, 3, 9, 27, 1, 4, 16, 5, 25, 125, ...

Sirul este obținut din sirul numerelor naturale prin încadrarea fiecărui număr par  $a$  cu secvența  $(1, a, a^2)$ , și înlocuirea fiecărui număr impar  $b$  cu secvența  $(b, b^2, b^3)$ .

## 2.4. Teste grilă



2.4.1. Care dintre secvențele de program de mai jos afișează corect elementele  $v[0]$  ,  $v[1]$  ,  $v[2]$ , . . . ,  $v[n-1]$  ale unui vector de întregi ?

a)  
i=0;  
while(i<n)  
{  
    cout<<v[i];  
    i++;  
}

b)  
i=0;  
while(i<n)  
{  
    i++;  
    cout<<v[i];  
}

c)  
i=0;  
do  
{  
    i++;  
    cout<<v[i];  
}while(i<n);

d)  
i=0;  
do  
{  
    cout<<v[i];  
    i++;  
}while(i<n);

e) nici una;



2.4.2. Care dintre secvențele de program de mai jos afișează corect produsul elementelor pare ale unui vector  $v = \{v[1], v[2], \dots, v[n]\}$  cu  $n$  elemente variabile de tip întreg ?

a)  
p=1;  
for(i=1;i<=n;i++)  
    if(v[i] % 2 == 0)  
        p=p\*v[i];  
cout << p;

b)  
p=1;  
for(i=0;i<n;i++)  
    if(v[i] / 2 == 0)  
        p=p\*v[i];  
cout << p;

c)  
p=0;  
for(i=0;i<n;i++)  
    if(v[i] % 2 != 0)  
        p=p\*v[i];  
cout << p;

d)

```
p=1;
for(i=0;i<n;i++)
    if(v[i] % 10 == 0)
        p=p*v[i];
cout << p;
```

e)

```
p=0;
for(i=0;i<n;i++)
    if(v[i] / 10 == 0)
        p=p*v[i];
cout << p;
```



2.4.3. Care dintre afirmațiile de mai jos sunt adevărate pentru secvența de program următoare ?

```
p=0;
for(k=1; k<6; k++)
    if(v[k]>v[p])
        p=k;
cout << p;
```

- a) Secvența este corectă din punct de vedere sintactic.
- b) Ciclul for are cinci pași.
- c) Dacă elementele vectorului ( $v[0]$  , $v[1]$ ,...,  $v[5]$ ) sunt 5,4,-11,9,-12,1 atunci programul afișează valoarea 4.
- d) Dacă elementele vectorului ( $v[0]$  , $v[1]$ ,...,  $v[5]$ ) sunt 3,-2,8,6,11,4, atunci programul afișează valoarea 4.
- e) Indiferent care ar fi elementele vectorului, secvența dată nu poate afișa valoarea 0.



2.4.4. Pentru secvența de program următoare, precizați care dintre afirmațiile de mai jos sunt adevărate.

```
int i=0, v[5]={1,1,1,1,1};
while( i<5 && v[i] && !v[i])
{
    v[i]=0;
    i++;
}
```

- a) Expresia din linia while este eronată sintactic.
- b) Declararea și inițializarea vectorului sunt corecte.
- c) După execuția secvenței toate elementele vectorului vor fi 1.
- d) După execuția secvenței toate elementele vectorului vor fi 0.
- e) Execuția secvenței va produce un ciclu infinit.



**2.4.5.** Precizați care va fi efectul secvenței de program următoare, în care  $v = \{ v[0], v[1], \dots, v[n-1] \}$  este un vector cu  $n$  elemente întregi.

```
x = v[n-1];
for(k=n-1; k>0; k--)
    v[k] = v[k-1];
v[0] = x;
```

- a) Deplasează toate elementele vectorului cu o poziție la dreapta
- b) Deplasează toate elementele vectorului cu o poziție la stânga.
- c) Șterge un element din vector prin deplasarea celor aflate înaintea lui.
- d) Rotește circular vectorul cu o poziție.
- e) Nici una dintre variantele anterioare.



**2.4.6.** Fie secvența de program următoare, în care  $v$  este un vector cu  $n$  elemente întregi, iar  $p$  este o variabilă de tip întreg:

```
for(p=1, k=1; k<0; k++)
    if( v[k] == v[k-1] )
        p = 0;
cout << p;
```

Secvența afișează 0 dacă:

- a) Toate elementele sunt distințe două câte două.
- b) Toate elementele sunt egale.
- c) Există două elemente consecutive distințe.
- d) Există două elemente consecutive egale.
- e) Nici una dintre variantele de mai sus.



**2.4.7.** Ce valoare va fi afișată în urma execuției programului următor?

- a) 0
- b) 1
- c) 3
- d) 5

e) programul va intra într-un ciclu infinit

```
#include<iostream.h>
int main()
{
    int v[]={0,1,2,0,4,5,6};
    int i=0, nr=0;
    do{
        if(i==v[i]) nr++;
    }while( i<6 && v[i++]);
    cout << nr;
}
```



**2.4.8.** Deducreți care vor fi, în ordine, de la stânga la dreapta, elementele nenule ale vectorului a la sfârșitul execuției secvenței de program următoare:

- a) 2,4,8
- b) 7,5
- c) 2,8
- d) 2,3,8
- e) 7,3,5

```
int i,j;
int v[7] = {0,2,7,3,4,8,5};
int a[7] = {0,0,0,0,0,0,0};
for (i=0; i<7; i++)
    if(( v[i] % 2 == 0 ) && ( i % 2 != 0 ))
    {
        a[j] = v[i];
        j++;
    }
```



**2.4.9.** Se consideră secvența următoare, în care valorile lui n și x se presupun cunoscute, iar v este un vector cu elementele ( $v[0]$ ,  $v[1]$ , ...,  $v[n-1]$ ).

```
p = -n;
for(i=0; i<n; i++)
    if(v[i] == x)
        p=x;
for(i=p+1; i<n; i++)
    v[i-1]=v[i];
for(i=0; i<n-1; i++)
    printf("%2d", v[i]);
```

Precizați care dintre următoarele afirmații sunt adevărate:

- a) Pentru  $n=5$ ,  $x=3$  și  $v=(5,6,2,7,1)$ , se afișează primele patru elemente nemodificate ale vectorului: 5 6 2 7
- b) Pentru  $n=5$ ,  $x=1$  și  $v=(2,i,3,i,4)$ , se afișează 2 3 1 4
- c) Secvența conține erori de sintaxă
- d) Algoritmul șterge din vector elementul cu valoarea x, prin mutarea cu o poziție mai la dreapta a elementelor aflate înaintea lui
- e) Algoritmul șterge din vector elementul cu valoarea x, prin mutarea cu o poziție mai la stânga a elementelor aflate după el



**2.4.10.** Declarați o matrice a cu 15 linii \* 10 coloane și elemente de tip întreg.

- a) int a[15][10];
- b) int a[15,10];
- c) a[15][10] int;
- d) int a[10][15];
- e) int a[10,15];



2.4.11. Care va fi matricea a în urma execuției secvenței de program de mai jos?

a) 
$$\begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix}$$

b) 
$$\begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}$$

c) 
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

d) 
$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

e) 
$$\begin{pmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \\ 1 & 1 & 1 \end{pmatrix}$$

```
n = 3;
for(i=0; i<n; i++)
    a[i][i] = 0;
for(i=0; i<n; i++)
    for(j=i+1; j<n; j++)
    {
        a[i][j] = 1;
        a[j][i] = -1;
    }
```



2.4.12. În secvența de program următoare, a este o matrice cu m linii și n coloane cu elemente de tip întreg, variabilele s și nr sunt de tipul int, iar m este de tipul float.

```
s=3; nr=0;
for(i=0;i<m;i++)
    for(j=0;j<n;j++)
        if( ......... )
        {
            s+=a[i][j];
            nr++;
        }
m=(float) s / nr;
cout << "m=" <<m;
```

Scriți condiția din linia if astfel încât secvența să afișeze corect media aritmetică a elementelor pozitive aflate pe linii pare și coloane impare în matrice.

- a)  $(a[i][j]>0) \&\& (i \% 2 == 0) \&\& (j \% 2 != 0)$
- b)  $(a[i][j]>0) \&\& ((i+j) \% 2 != 0)$
- c)  $(a[i][j]>0) \&\& (i/2 == 0) \&\& (j \% 2 != 0)$
- d)  $(a[i][j]>0) \&\& (i/2 != 0) \&\& (j \% 2 == 0)$
- e) Nici una din condițiile anterioare



2.4.13. Fie o matrice a cu m linii \* n coloane și elemente întregi. Care dintre secvențele de program de mai jos memorează în vectorul  $b = (b[0], b[1], \dots, b[m-1])$  elementele maxime de pe liniile matricei? (adică  $b[i]$  să fie maximul de pe linia  $i$  a matricei, cu  $i=0,1,2,\dots,m-1$ )

a)  
`for(i=0;i<m;i++)  
{  
 x=a[i][0];  
 for(j=1;j<n;j++)  
 if(a[i][j]>x)  
 x=a[i][j];  
 b[i]=x;  
}`

b)  
`for(i=0;i<m;i++)  
{  
 x=a[i][n-1];  
 for(j=1;j<n-1;j++)  
 if(a[i][j]>x)  
 x=a[i][j];  
 b[i]=x;  
}`

c)  
`for(i=0;i<m;i++)  
{  
 x=a[i][n-1];  
 for(j=n-2;j>=0;j--)  
 if(a[i][j]>x)  
 x=a[i][j];  
 b[i]=x;  
}`

d)  
`for(i=0;i<m;i++)  
{  
 x=-MAXINT;  
 for(j=0;j<n;j++)  
 if(a[i][j]>x)  
 x=a[i][j];  
 b[i]=x;  
}`

e) toate secvențele anterioare



2.4.14. Se consideră cele două secvențe următoare, în care a este o matrice de întregi cu m linii \* n coloane, iar x este o variabilă de tip întreg. Valorile lui m, n, x, precum și elementele matricei se presupun cunoscute (citite anterior).

// Secventa S1  
`for(j=x+1; j<=n-1; j++)  
 for(i=0; i<=m-1; i++)  
 a[i][j-1]=a[i][j];`

// Secventa S2  
`for(j=n-1; j>=x+1; j--)  
 for(i=m-1; i>=0; i--)  
 a[i][j-1]=a[i][j];`

Precizați care dintre următoarele afirmații sunt adevărate:

a) Cele două secvențe sunt echivalente (produc același efect în orice situație).

b) Dacă se execută secvența S1 pentru  $x=1$ ,  $m=3$ ,  $n=4$  și matricea

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

atunci după execuția secvenței matricea va fi

$$\begin{pmatrix} 1 & 3 & 4 \\ 5 & 7 & 8 \\ 9 & 11 & 12 \end{pmatrix}$$

c) Dacă se execută secvența S1 pentru  $x=4$ ,  $m=3$ ,  $n=4$  și matricea

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

atunci după execuția secvenței matricea va fi

$$\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & 10 & 11 \end{pmatrix}$$

d) În secvența S2 ciclurile for sunt eronate din punct de vedere sintactic.

e) Nici una dintre afirmațiile de mai sus



**2.4.15.** Fie secvența de program, în care lipsesc elementele vectorului v la inițializare.

```
int i=0, v[6]={ . . . . }, s;
for(s=5, i=1; i<6; s=v[i]-v[i-1], i++);
cout << s;
```

Ultima valoare afișată va fi 0, dacă:

- a) Primul element al vectorului este 5, iar celelalte elemente sunt nule
- b) Ultimul element al vectorului este 5, iar celelalte elemente sunt nule
- c) Fiecare element, începând cu al doilea, este mai mare cu 1 decât elementul aflat înaintea lui în vector
- d) Fiecare element, începând cu al doilea, este mai mic cu 1 decât elementul aflat înaintea lui în vector
- e) Toate elementele vectorului au valoarea 5



**2.4.16.** Se consideră vectorii  $u$  și  $v$ , cu  $m$ , respectiv,  $n$  elemente de tip întreg.

Precizați care dintre situațiile descrise mai jos sunt adevărate în urma execuției secvenței alăturate.

```
k=0;
for(i=0,i<m;i++)
    for(j=0,j<n;j++)
        if(u[i]==v[i])
            w[k++]=u[i];
```

- c) Pentru vectorii  $u = (2, -3, 1)$  cu  $m=3$  elemente și  $v = (-3, 4, 2, 5)$  cu  $n=4$  elemente, se va obține în urma execuției secvenței vectorul  $w = (2, -3)$
- d) Pentru vectorii  $u = (3, 7, 3)$  cu  $m=3$  elemente și  $v = (2, 7, 2, 3)$  cu  $n=4$  elemente, se va obține în urma execuției secvenței vectorul  $w = (3, 7)$
- e) Pentru vectorii  $u = (1, -2, 6)$  cu  $m = 3$  elemente și  $v = (3, 4, 8, 9)$  cu  $n=4$  elemente, la finele execuției secvenței date vectorul  $w$  nu va avea nici un element
- f) Dacă înaintea execuției secvenței unul dintre vectorii  $u$ ,  $v$  nu conține nici un element, atunci după execuția secvenței vectorul  $w$  nu va conține nici un element



**2.4.17.** Precizați care atribuiriri sunt corecte în programul următor:

- a) I
- b) II
- c) III
- d) IV
- e) nici una

```
int main()
{
    int *a[10], (*b)[10];
    int x, y[10];
    a[0]=&x;      // (I)
    b=&y;        // (II)
    a[4]=y;       // (III)
    b=&y[4];     // (IV)
}
```



**2.4.18.** Care dintre instrucțiunile de tipărire din programul de mai

jos:

- a) I și II
- b) I și IV
- c) II și III
- d) III și IV
- e) nici una

```
#include<stdio.h>
int main()
{
    int **a[10], *b[10], c[10];
    b[3]=&c[5];
    a[4]=&b[3];
    c[5]=3;
    printf("%d\n",**a[4]);           // (I)
    printf("%d\n",*(a+4));          // (II)
    printf("%d\n",*(a+4));          // (III)
    printf("%d\n",***a[4]);         // (IV)
}
```



**2.4.19.** Precizați ce valoare va afișa programul următor:

- a) Adresa lui b
- b) Adresa lui a[2]
- c) 12
- d) NULL
- e) Programul are erori de sintaxă

```
#include<stdio.h>
int main()
{
    int x=12;
    int *a[10], *&b=a[2];
    b=&x;
    printf("%d\n",*a[2]);
}
```



**2.4.20.** Fie vectorul y cu patru elemente numere întregi:

```
int y[4]={0,1,2,3};
```

Care dintre următoarele instrucțiuni declară și inițializează corect un pointer ptr către vectorul y ?

- a) int \*(ptr[4])=&y;
- b) int (ptr\*) [4]=&y;
- c) int (\*ptr) [4]=&y;
- d) int ptr\* [ 4 ] =&y;
- e) int \*ptr [ 4 ] = &y;



**2.4.21.** Precizați care dintre situațiile de mai jos sunt adevărate în timpul execuției programului dat.

```

#include<iostream.h>
int main()
{
    int m[20], i, n=10, s;
    m[0]=m[1]=1 ;
    for(i=2; i<n; i++)
        m[i]=m[i-1] + i%3;
    for(s=*m; i=1; i<n; i++)
        s=( *(m+i) != *(m+i-1) ) ? s+*(m+i) : s;
    cout << s;
}

```

- a) Elementele vectorului m sunt toate distincte între ele, cu excepția primelor două
- b) Programul afișează valoarea 30
- c) Expresia condițională din al doilea ciclu for este eronată
- d) Programul conține erori legate de referirea vectorului m prin intermediul pointerilor
- e) Nici una dintre afirmațiile de mai sus



2.4.22. De câte ori va afișa valoarea 30 programul următor ?

- a) nici o dată
- b) o dată
- c) de două ori
- d) de trei ori
- e) Programul generează eroarea "Divide error" din cauza unei împărțiri la 0.

```

#include<iostream.h>
int main()
{
    int *a[5], i=0, v[5], x=30;
    while(i<5) v[i]=x/i;
    for(i=1; i<5; i++)
        *(a+i)=&v[i];
    a[0]=&x;
    for(i=0; i<5; i++)
        cout << "\n" << *a[i];
}

```



2.4.23. Ce valoare va afișa programul următor?

- a) 10
- b) 6
- c) 3
- d) -3
- e) programul este eronat

```
#include<iostream.h>
int main()
{
    int a[20][20], i, j, n=4;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            *(*(a+i)+j) = ( i>j ) ? ( j-i ) : ( j+i );
    int m=10;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            if ( m > (*(a+i)) [j] )
                m=a[i][j];
    cout << m << "\n";
}
```



**2.4.24.** Fie următorul program:

```
#include<stdio.h>
int main()
{
    int u[4] = {1,2,3,4}, v[4] = {5,6,7,8}, w[4] = {0,0,0,0} i;
    int (*x)[4] = &u, (*y)[4] = &v, (*z)[4] = &w;
    for(i=0; i<4; i++)
        printf("%3d", (*z)[i] = (*x)[i] + (*y)[i]);
}
```

Care dintre afirmațiile de mai jos sunt adevărate?

- a) Programul va afișa patru adrese de memorie
- b) Programul va afișa, în ordine, valorile 6,8,10,12
- c) Valoarea lui  $(*x)[2]$  este 3
- d) Valoarea lui  $(*y)[4]$  este 8
- e) Instrucțiunea de afișare din ciclu este eronată, din cauza folosirii operatorului de atribuire în funcția printf



2.4.25. Fie următorul program:

```
#include<iostream.h>
int main()
{
    int x[4]={1,2,3}, y[4]={4,5,6,7}, z[7], i,j;
    for(i=0; i<4; i++)
        *(z+i) = *(y+i);
    for(j=0; j<3; i++)
        *(z+i+j) = *(x+j);
    for(i=0; i<7; i++)
        cout << *(z+i);
}
```

Care vor fi valorile afişate în urma execuției sale ?

- a) 1,2,3,4,5,6,7
- b) 7,6,5,4,3,2,1
- c) 3,2,1,7,6,5,4
- d) 4,5,6,7,1,2,3
- e) Programul este eronat

## 2.5. Răspunsuri la întrebările grilă

2.5.1. a) și d)

2.5.2. a)

2.5.3. a), b) și d)

2.5.4. b) și c)

2.5.5. d)

2.5.6. b) și d)

2.5.7. b)

2.5.8. c)

2.5.9. e)

2.5.10. a)

2.5.11. b)

2.5.12. a), c) și d)

2.5.13. e)

2.5.14. b)

2.5.15. b) și c)

2.5.16. a), c) și d)

2.5.17. a), b) și c)

2.5.18. a) și b)

2.5.19. c)

2.5.20. c)

2.5.21. b)

2.5.22. c)

2.5.23. d)

2.5.24. b) și c)

2.5.25. d)

# Şiruri de caractere. Structuri şi uniuni

## 3.1. Şiruri de caractere

**D**atele care se reprezintă sub formă de şiruri de caractere au o largă aplicabilitate în programarea calculatoarelor, indiferent de limbajul folosit. Astfel și în limbajul C++ se pot memora și prelucra informații de tip şir de caractere. Cu toate că limbajul C++ nu conține un tip de date special pentru şiruri de caractere aşa cum are limbajul Pascal, se pot utiliza tablouri unidimensionale de caractere.

Declararea unui tablou de caractere se face astfel:



**char num\_tablou[dimensiune\_maxima];**



Exemple:

```
char sir[20]; // tablou de 20 de caractere  
char t[10]; // tablou de 10 caractere
```

Pentru a specifica sfârșitul şirului de caractere, după ultimul caracter se adaugă un octet cu valoarea 0 ( caracterul '\0' ).

Dimensiunea declarată pentru un tablou de şiruri de caractere trebuie să fie cu o unitate mai mare pentru ca pe ultima poziție să se poată pune și valoarea '\0' – terminatorul de şir.

Reprezentarea internă a unui sir de caractere



**char sir[33] = "Limbajul C-teorie si aplicatii";**

Acest sir va fi memorat astfel:

L	i	m	b	a	j	u	l	C	-	t	e	o	r	i	e	s	i	a	p	l	i	c	a	t	i	i	\0			
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	--	--	--

sir[0]      sir[33] = "Limbajul C-teorie si aplicatii"

Ultimi octeți sunt nefolosiți.

În limbajul C, un sir de caractere este un tablou unidimensional cu elemente de tip caracter și care se termină cu NULL. Totuși compilatorul C nu adaugă automat terminatorul NULL, decât în cazul folosirii funcțiilor predefinite *fgets()* și *gets()*, iar în celealte cazuri este necesar ca programatorul să adauge terminatorul de sir atunci când dorește acest lucru, pentru a lucra cu sirurile de caractere.

Exemplu: Următorul program declară un sir de caractere de 256 de elemente și atribuie primelor 26 de locații libere literele mari ale alfabetului:



```
#include<stdio.h>
int main(void)
{
    char sir[256];
    int i;
    for(i=0;i<26;i++)
        sir[i] = 'A' + i;
    sir[i] = NULL;
    printf("Sirul de caractere contine: %s\n",sir);
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

Sirul de caractere contine: ABCDEFGHIJKLMNOPQRSTUVWXYZ

În biblioteca limbajului C++ există câteva funcții specifice șirurilor de caractere:

- în fișierul standard de intrare / ieșire – *stdio.h*, avem funcțiile *gets()* și *puts()*.
- în fișierul *string.h*, avem mai multe funcții pe care le vom prezenta mai departe.

Deocamdată, vom prezenta câteva funcții construite astfel încât să exemplificăm mai ușor funcționarea celor predefinite.

### Problema 1:

- a) Să se construiască o funcție care să returneze lungimea unui șir de caractere dat ca parametru.
- b) Să se construiască o funcție care să copieze un șir în alt șir.
- c) Să se construiască o funcție care să concateneze două șiruri de caractere.
- d) Să se construiască o funcție care să caute un șir de caractere în alt șir de caractere.



```
#include<stdio.h>
// determinarea lungimi unui sir
int unsigned lungime(char a[])
{
    int i = 0;
    while( a[i] != NULL ) i++;
    return i;
}
// copierea unui sir in alt sir
char copiaza(char dest[], char sursa[])
{
    int i = 0;
    while( sursa[i] != NULL )
    {
        dest[i] = sursa[i];
        i++;
    }
    dest[i]=NULL;
}
```

```

// concatenarea a doua siruri
char concatenate(char a[], char b[])
{
    int i=0,j=0;
    while(a[i]!=NULL) i++;
    while(b[j]!=NULL)
    { a[i]=b[j];j++;i++; }
    a[i]=NULL;
}

// cauta un subsir in altul
int cauta_sir(char sir[], char subsir[])
{
    int i,j,k,nr=0;
    for(i=0;sir[i];i++)
        for(j=i, k=0; sir[j]==subsr[k]; j++,k++)
            if(!subsr[k+1]) nr++;
    return nr;
}

int main(void)
{
    char sir[256],sir1[256],sir2[256];
    printf("\nDati sirul ");      gets(sir);
    printf("Sirul de caractere are %d caractere\n",
lungime(sir));
    printf("\nDati sirul sursa ");    gets(sir1);
    copiaza(sir2,sir1);
    printf("\nSirul destinatie este: %s",sir2);
    printf("\nDati sirul 1 ");    gets(sir1);
    printf("\nDati sirul 2 ");    gets(sir2);
    concatenate(sir1,sir2);
    printf("\nSirul concatenat este: %s",sir1);
    printf("\nDati sirul ");      gets(sir);
    printf("\nDati subsirul care se cauta ");  gets(sir1);
    printf("Subsirul dat apare de %d ori",
cauta_sir(sir,sir1));
}

```

### Problema 2:

Să se construiască o funcție care să șteargă un subșir de caractere dintr-un alt sir de caractere.



```
#include<stdio.h>
char *stergere(char *sir, char *subsr)
{
    int i,j,k,poz=-1;
    // cauta subsirul in sirul dat si retine pozitia de unde incepe
    for(i=0;(sir[i]) && (poz == -1);i++)
        for(j=i, k=0; sir[j]==subsr[k]; j++,k++)
            if(!subsr[k+1]) poz=i;
    if(poz!=-1) // daca subsirul a fost gasit
    {
        for(k=0;subsr[k];k++) ; //numara cate caractere are
        subsirul
        for(j=poz, i=poz+k; sir[i]; j++, i++) sir[j]=sir[i];
        sir[j]=NULL;
    }
    return sir;
}
int main(void)
{
    char sir[256],sir1[256];
    printf("\nDati sirul "); gets(sir);
    printf("\nDati subsirul care se sterge "); gets(sir1);
    printf("Sirul ramas dupa stergere este %s",
    stergere(sir,sir1));
}
```

### Problema 3:

Următorul program ilustrează modul de folosire a funcțiilor de conversie dintr-un sir de caractere într-un număr. Pentru aceasta, mai întâi prezentăm în tabelul următor aceste funcții:



<b>FUNCȚIE</b>	<b>LA CE FOLOSEȘTE</b>
<b>atof</b>	convertește un sir de caractere într-un număr real simplă precizie
<b>atoi</b>	convertește un sir de caractere într-un număr întreg
<b>atol</b>	convertește un sir de caractere într-un număr întreg de tip long
<b>strtod</b>	convertește un sir de caractere într-un număr real dublă precizie
<b>strtol</b>	convertește un sir de caractere într-un număr de tip long



```
#include<stdio.h>
#include<stdlib.h>
int main(void)
{
    int numar_int;
    float numar_real;
    long numar;
    numar_int = atoi("6789");
    numar_real = atof("12.345");
    numar = atol("1234567890L");
    printf("%d %f %ld\n", numar_int, numar_real, numar);
}
```

#### Problema 4:

Următorul program ilustrează modul de folosire a funcțiilor de conversie dintr-un număr într-un sir de caractere. Pentru aceasta, mai întâi prezentăm în tabelul următor, aceste funcții:



#### FUNCȚIE

**itoa**

convertește un număr întreg într-un sir de caractere

**ftoa**

convertește un număr real simplă precizie într-un sir de caractere

**ultoa**

convertește un număr de tip unsigned long într-un sir de caractere

#### LA CE FOLOSEȘTE



```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
int main(void)
```

```
{
```

```
    int numar_int=6789;
```

```
    float numar_real;
```

```
    long numar=1234567890L;
```

```
    char sir[25];
```

```
    itoa(numar_int, sir, 10);
```

```
    printf(" numar = %d sir = %s\n", numar_int, sir);
```

```
    ltoa(numar, sir, 10);
```

```
    printf(" numar = %ld sir = %s\n", numar, sir);
```

```
}
```

## 3.2. Probleme rezolvate

### 3.2.1. Enunțuri

1. Să se scrie o funcție care să returneze lungimea unui sir de caractere, transmis ca parametru.
2. Scrieți un program care citește un sir de caractere și transformă sirul în sir cu litere mici.
3. Să se afișeze unul sub altul, toate prefixele unui cuvânt citit de la tastatură (prefixele unui cuvânt sunt compuse din minim un caracter și maxim toate caracterele, citite de la stânga la dreapta).
4. Să se verifice dacă două cuvinte citite de la tastatură rimează (spunem că două cuvinte rimează dacă ultimele două caractere sunt identice).
5. Se citește un sir de caractere de la tastatură. Să se scrie un program C/C++ care să afișeze numărul de vocale și de consoane din sirul dat.

### 3.2.2. Soluții propuse

#### Problema 1:

Să se scrie o funcție care să returneze lungimea unui sir de caractere, transmis ca parametru.



```
#include<iostream.h>
int lungime(char s[30])
{
    int i;
    i=0;
    while(s[i]!=0) i++;
    return i;
}
int main(void)
{
    char s[30];
    int l;
    cout<<"Dati sirul de caractere ";cin>>s;
    l=lungime(s);
    cout<<"Numarul de caractere ale sirului este "<<l;
}
```

### Problema 2:

Scrieți un program care citește un sir de caractere și transformă sirul în sir cu litere mici.



```
#include<iostream.h>
void litere_mici(char s[30])
{
    int i;
    i=0;
    while(s[i]!=0)
    {
        if( (s[i]>=65) && (s[i]<=90) )      s[i]=s[i]+32;
        i++;
    }
    return;
}
int main(void)
{
    char s[30];
    int l;
    cout<<"Dati sirul de caractere ";cin>>s;
    litere_mici(s);
    cout<<"Sirul transformat in litere mici este "<<s;
}
```

### Problema 3:

Să se afișeze unul sub altul, toate prefixele ale unui cuvânt citit de la tastatură (prefixele unui cuvânt sunt compuse din minim un caracter și maxim toate caracterele, citite de la stânga la dreapta).



```
#include<iostream.h>
void prefixe(char s[30])
{
    int i,j;
    i=0;
    while(s[i]!=0)
    {
        for(j=0;j<=i;j++) cout<<s[j];
        cout<<endl;
        i++;
    }
    return;
}
```

```

int main(void)
{
    char s[30];
    int l;
    cout<<"Dati sirul de caractere ";cin>>s;
    cout<<"Prefixele sirului dat sunt : "<<endl;
    prefixe(s);
}

```

#### Problema 4:

Să se verifice dacă două cuvinte citite de la tastatură rimează (spunem că două cuvinte rimează dacă ultimele două caractere sunt identice).



```

#include<iostream.h>
int rima(char s[30],char t[30])
{
    int i,j;
    i=0;
    while(s[i]!=0) i++;
    j=0;
    while(t[j]!=0) j++;
    if( (s[i]==t[j]) && (s[i-1]==t[j-1]) ) return 1;
    else return 0;
}
int main(void)
{
    char s[30],t[30];
    int l;
    cout<<"Dati primul sir de caractere ";cin>>s;
    cout<<"Dati al doilea sir de caractere ";cin>>t;
    if( rima(s,t)==1 ) cout<<s<<" rimeaza cu "<<t;
    else cout<<s<<" NU rimeaza cu "<<t;
}

```

#### Problema 5:

Se citește un sir de caractere de la tastatură. Să se scrie un program C/C++ care să afișeze numărul de vocale și de consoane din sirul dat.



```
#include<iostream.h>
int main(void)
{
    char s[50];
    int voc=0,cons=0,i;
    cout<<"Dati sirul de caractere ";cin>>s;
    i=0;
    while(s[i]!=0)
    {
        switch (s[i])
        {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
            case 'A':
            case 'E':
            case 'I':
            case 'O':
            case 'U': {voc++;break;}
            default : cons++;
        }
        i++;
    }
    cout<<"Numarul de vocale ale sirului este
"<<voc<<endl;
    cout<<"Numarul de consoane ale sirului este
"<<cons<<endl;
}
```

### 3.3. Probleme propuse spre rezolvare

1. Să se afișeze toate prefixele și toate sufixele unui cuvânt citit de la tastatură.

*Exemplu :* Prefixele cuvântului ‘informatică’ sunt: ‘i’, ‘in’, ‘inf’, ‘info’, ‘infor’, ‘inform’, ‘informa’, ‘informat’, ‘informati’, ‘informatic’, ‘informatică’ iar Sufixe sunt : ‘informatică’, ‘informatică’ ‘nformatică’, ‘formatică’, ‘ormatică’, ‘rmatică’, ‘matică’, ‘atică’, ‘tică’, ‘ică’, ‘că’, ‘ă’.

2. Se citește un sir de caractere (litere mari și mici) de la tastatură. Se cere :

- Să se afișeze sirul de caractere doar cu litere mari.
- Să se afișeze sirul de caractere doar cu litere mici.

*Exemplu :* Sirul ‘inFORrmATica’ va deveni

- ‘INFORMATICA’ și respectiv
- ‘informatica’.

3. De pe un rând al ecranului se citește un text care poate să conțină orice caractere (litere mari, litere mici, cifre și caractere speciale). Să se tipărească textul obținut prin transformarea în litere mari a literelor mici aflate pe poziții impare în cadrul textului dat (prima, a treia, a cincea etc.).

*Exemplu:* Sirul 'AbcdeFGHijkIMNOPRstu' devine 'AbCdEFGHIkLMNOPRsTU'.

4. Se citește de la tastatură un caracter c și un text de maxim 70 caractere. Textul este citit pe un singur rând de ecran. Afisați de câte ori apare caracterul c în cadrul textului. Literele mici nu se vor considera diferite de majuscule.

*Exemplu :* În textul "Acesta este un exemplu destul de banal" litera 'a' apare de 4 ori.

**5.** Se dă un sir de caractere s și o mulțime A de caractere. Să se determine câte din elementele mulțimii A apar în scrierea lui s.

*Exemplu :* Dacă s='Elefantul are trompa dar iepurele are urechi lungi' iar A = ('a', 'x', 'g', 'e', 'v') atunci se vor afișa literele 'a', 'g' și 'e'.

**6.** Să se eliminate toate aparițiile caracterului spațiu dintr-un text.

*Exemplu :* Dacă textul inițial este ' abc def ghi ' în final se va obține 'abcdefghi'.

**7.** Se citește un text de la tastatură. Să se eliminate spațiile inutile (caracterul spațiu înainte de unul din caracterele ',', ';', ',' sau mai mult de un spațiu între două cuvinte, sau spațiile de la începutul și de la sfârșitul textului). După eliminare să se tipărească textul obținut.

*Exemplu :* Textul ' abc def , ghi ' devine 'abc def, ghi'.

**8.** Să se scrie un program care citește mai multe linii de la tastatură și o tipărește pe cea mai lungă. Citirea de la tastatură se termină când s-a introdus o linie goală.

*Exemplu :* Dacă liniile citite de la tastatură sunt:

Elevul ieșe la tabla  
El raspunde la intrebarile profesorului  
Daca raspunde corect ia nota mare

se va afișa cea de-a doua linie care este cea mai lungă.

**9.** Se citește de la tastatură un sir de caractere care poate să conțină numai litere și cifre. Afisati numărul literelor mari, numărul literelor mici și numărul caracterelor-cifră din text.

*Exemplu :* În textul ‘S-a nascut la Bucuresti in 1874 pe 4 mai. A trait mai bine de 80 de ani.’ Se găsesc 43 de litere mici, 3 litere mari și 7 cifre.

10. Se citesc de la tastatură mai multe linii. Scrieți un program care să tipărească toate liniile mai scurte de 80 de caractere. Citirea de la tastatură se termină când s-a introdus o linie goală.

11. Să se verifice dacă două cuvinte introduse de la tastatură sunt *rime* (dacă ultimele p caractere ale celor două cuvinte coincid, unde p este o valoare dată).

*Exemplu :* Dacă p=2 atunci cuvintele ‘acasa’ și ‘masa’ sunt rime.

12. Se citește de la tastatură o frază de lungime cel mult 70 de caractere. Între cuvintele frazei pot apărea ca separatori numai spațul, virgula și punctul. Să se afișeze unul sub altul cuvintele frazei în ordine alfabetică.

*Exemplu :* Pentru fraza "Acesta este un exemplu simplu, dar bun, de fraza ." se vor afișa cuvintele în următoarea ordine: ‘Acesta’, ‘bun’, ‘dar’, ‘de’, ‘este’, ‘exemplu’, ‘fraza’, ‘simplu’, ‘un’.

13. Se citește de la tastatură un text. Se știe că două cuvinte din text sunt separate prin unul sau mai multe spații. Să se afișeze toate cuvintele din text formate din k caractere, unde k este un număr natural dat.

*Exemplu :* Considerăm k=2 și textul ‘Mergeam la plimbare și vorbeam cu prietena mea’. Aceasta conține 3 cuvinte de lungime 2.

14. Scrieți un program care convertește un sir de caractere într-un întreg. Conversia se oprește odată cu întâlnirea unui caracter ce nu este cifră.

*Exemplu :* Dacă sirul este '123ef34' se va obține numărul întreg 123,

15. Se citește de la tastatură o frază de maxim 70 de caractere. Să se afișeze, una sub alta, toate perechile de vocale situate pe poziții consecutive în frază, precum și numărul acestora.

*Exemplu* : Pentru textul ‘aceasta bluza desi foarte ieftina foarte frumoasa asa ca am cumparat-o’, are 5 perechi de vocale consecutive și anume: ‘ea’, ‘oa’, ‘ie’, ‘oa’, ‘oa’.

16. Se citește un text T de la tastatură și două cuvinte s1 și s2. Să se scrie un program care tipărește textul obținut din T prin înlocuirea cuvântului s1 cu cuvântul s2. Cuvintele se consideră separate prin unul sau mai multe mai multe caractere ‘,’ , ‘.’ , ‘;’ și spațiu.

*Exemplu* : Dacă în textul ‘Ea este eleva. El este student și mi-a dat o veste foarte buna’ se înlocuiește cuvântul ‘este’ cu ‘era’ se obține textul ‘Ea era eleva. El era student și mi-a dat veste foarte buna.’

17. Se dau două siruri de caractere de lungimi egale. Al doilea sir conține doar cifre, în caz contrar programul terminându-se fără nici o prelucrare. Să se construiască un al treilea sir, prin repetarea pe rând, a fiecărui caracter din primul sir, de un număr de ori egal cu cifra corespunzătoare din al doilea sir.

*Exemplu*: ‘info’ și ‘4352’ => ‘iiiiinnnfffffoo’.

18. Un cuvânt este palindrom dacă citind literele de la dreapta la stânga obținem același cuvânt. Scrieți un program care verifică dacă un cuvânt citit de la tastatură este palindrom sau nu.

*Exemplu* : Cuvintele ‘capac’ și ‘lupul’ sunt palindroame.

19. Se citește de la tastatură un cuvânt de lungime de cel mult 20 de caractere format numai din litere mari. Să se afișeze toate cuvintele distincte ce se pot forma prin eliminarea câte unui singur caracter din cuvântul dat.

*Exemplu:* pentru cuvântul STUDENT se vor afișa, nu neapărat în această ordine, cuvintele: TUDENT, SUDENT, STDENT, STUENT, STUDNT, STUDET, STUDEN.

**20.** Să se scrie un program care citește un text de la tastatură și îl traduce în limba ‘păsărească’ astfel: fiecare vocală  $\beta$  va fi înlocuită prin sirul  $\beta p\beta$ .

*Exemplu :* Pentru sirul ‘pepene rosu’ obținem ‘pepepepenepenepoșupu’.

**21.** Se citesc de la tastatură n siruri de caractere care pot conține numai litere și cifre. Să se afișeze câte dintre sirurile citite nu pot fi transformate în numere întregi, precum și suma numerelor rezultate prin transformarea sirurilor care pot fi convertite.

*Exemplu :* Dacă sirurile citite sunt ('451', '2', '3', '254', 'cdg', 'info', '22', '10') atunci 3 siruri nu pot fi transformate în numere întregi iar suma numerelor rezultate prin transformarea sirurilor care pot fi convertite este 737.

**22.** Să se scrie un program care tipărește distribuția frecvențelor lungimii cuvintelor aflate într-un text citit de la tastatură. Cuvintele sunt separate prin spații.

*Exemplu :* Pentru textul ‘Toamna aceasta ploua foarte tare desi nu prea imi place acest lucru’ se va afișa:

1 cuvânt de lungime 2  
1 cuvânt de lungime 3  
3 cuvinte de lungime 4  
4 cuvinte de lungime 5  
2 cuvinte de lungime 6  
1 cuvânt de lungime 7

**23.** Să se scrie un program care va realiza începerea fiecărei propoziții dintr-un text cu literă mare. Spațiile inutile din text se vor elimina. Dacă nu s-a facut

nici o modificare se va da un mesaj corespunzător. Propozițiile sunt despărțite între ele de un caracter punct urmat de zero, unul sau mai multe spații.

*Exemplu :* Textul ‘toamna a venit devreme. ploua si e frig, soarele nu mai arde asa de tare.’ se transforma în ‘Toamna a venit devreme. Ploua si e frig. Soarele nu mai arde asa de tare.’

**24.** Să se scrie un program care inversează într-un text toate cuvintele care încep cu litera ‘a’ (sau ‘A’) prin oglindire. Cuvintele din text se consideră separate prin spații (unul sau mai multe).

*Exemplu :* Textul ‘Aceasta este un exemplu aproape perfect asa ca am insistat sa-l vezi acum si tu.’ Se va transforma în ‘atsaecA este un exemplu epaorpa perfect asa ca ma insistat sa-l vezi muca si tu.’

**25.** Să se determine numărul cuvintelor dintr-un text dat care sunt mai lungi de 7 litere și conțin litera u. Se presupune că în textul dat cuvintele sunt alcătuite numai din litere și sunt separate prin spațiu sau virgulă.

*Exemplu :* Dacă textui este ‘Exemplul acesta nu este cu nimic superior, comparativ cu celealte, este doar un exemplu’ se va afișa valoarea 3.

**26.** Se dă un sir de caractere. Să se eliminate din acesta tot ce nu e literă sau cifră și să se afișeze în noua formă împreună cu lungimea anterioară și cea curentă.

*Exemplu :* Dacă sirul inițial este '#exemplu de 3 siruri ^^nice^^ !!! Nota 10' se obține sirul ‘exemplude3sirurinicenota10’. Lungimea sirului inițial este 41 iar a celui final 26.

**27.** Să se scrie un algoritm care multiplică un sir de caractere dat de n ori.

*Exemplu :* Fiind dat sirul 'INFO' si n=2, algoritmul va genera sirul 'INFOINFO'.

**28.** Se citesc de la tastatură două siruri de litere ale alfabetului englez. Să se tipărească vocalele care nu apar în nici unul din cele două siruri. Un caracter va fi afişat doar o dată. Literele mici nu se vor considera distincte de majuscule.

*Exemplu :* Dacă cele două siruri sunt 'Elicbpter' și 'TELEFON' atunci se vor afișa literele 'A', 'U'.

**29.** Se citesc de la tastatură două siruri de litere ale alfabetului englez. Să se tipărească caracterele care nu sunt comune celor două siruri (adică apar doar într-un singur sir). Un caracter va fi afişat doar o dată. Literele mici nu se vor considera distincte de majuscule. Nu se vor folosi alte tipuri structurate de date.

*Exemplu :* Dacă cele două siruri sunt 'Elicopter' și 'TELEFON' atunci se vor afișa literele 'c', 'F', 'I', 'N', 'P', 'R'.

**30.** Un text conține litere mari și una sau mai multe secvențe:

'\Punct'      '\Excl'      '\Inter'      '\Virg'

Să se rescrie textui înlocuind secvențele de mai sus cu

'.'      '!'      '?'      ','

**31.** Să se verifice dacă două cuvinte se pot obține unul din celălălt prin permutarea literelor. Nu se face distincție între literele mari și mici ale alfabetului.

*Exemplu :* Cuvântul 'CUPTOR' se poate obține din cuvântul 'CoRuPt' prin permutarea literelor.

**32.** Se citește de la tastatură un sir de caractere. Să se afișeze caracterul (caracterele) care apare (apar) de cele mai multe ori în componența sirului.

*Exemplu :* Dacă sirul este ‘akvkbuvkv’ , programul va tipări caracterele ‘k’ și ‘v’, care apar de câte trei ori fiecare.

**33.** Se citește de la tastatură un număr întreg cu maxim 10 cifre, sub forma unui sir de cifre. Să se eliminate o cifră aleasă astfel încât numărul rămas să aibă cifrele în ordine crescătoare. Dacă sunt mai multe soluții se vor afișa toate, iar dacă problema nu are nici o soluție se va tipări un mesaj. Un număr se va afișa o singură dată.

*Exemplu :* Pentru  $m=2435$ , poate fi eliminată cifra 3 rămânând numărul 245, sau cifra 4 rămânând numărul 235.

**34.** Să se determine toate numerele mai mici ca 10000 cu proprietatea că pătratul numărului este format din aceleași cifre ca și numărul.

*Exemplu :* Un astfel de număr este 4762 ( $4762^2=22676644$ ).

## 3.4. Structuri și uniuni

Există situații de programare în care folosirea variabilelor simple sau a tablourilor nu mai este eficientă deoarece variabilele, dar mai ales tablourile, nu permit decât prelucrarea unui anumit tip de date. Astfel a apărut necesitatea folosirii unui nou mecanism care să permită definirea și utilizarea de tipuri de date complexe, adică să permită memorarea la un moment dat a mai multor date, de diverse tipuri care pot fi prelucrate fie individual, fie ca o entitate. Acest lucru se poate face folosind tipul de date utilizator structură - struct.

O structură constă dintr-un număr oarecare de date care pot fi de același tip sau de tipuri diferite, grupate împreună.

### 3.4.1. Definirea structurilor



```
struct nume_structură
{
    elementele structurii
};
```

Exemplu: Definim, în continuare, o structură cu numele complex, care are doi membri, ce semnifică partea reală și partea imaginară a unui număr complex:



Exemplu:

```
struct complex
{
    double real;
    double imaginar;
};
```

### 3.4.2. Declararea variabilelor de tip structură



```
struct nume_structură  
listă_variabile;  
sau  
struct nume_structură  
{  
    elementele structurii  
}listă_variabile;
```

Exemplu: Definim, aceeași structură ca și mai înainte, dar și declarăm lista de variabile de tip structură:



Exemplu:  
**struct complex z1, z2;**  
sau  
**struct complex**  
{  
 double real,imaginar;  
}z1, z2;

### 3.4.3. Accesarea elementelor unei structuri

Pentru a avea acces la un element al unei structuri se folosește operatorul ‘.‘, operator membru. Acesta are rolul de a efectua conexiunea dintre un nume de variabilă de tip structură și un (membru) element al acesteia.

Exemplu: Pentru aceeași structură definită mai înainte, accesul la fiecare membru al structurii se face astfel:



Exemplu:  
**z1.real=0.1** – accesează primul element al structurii  
**z1.imaginar=0** – accesează al doilea element al structurii

### 3.4.4. Tablouri de structuri

Structurile pot fi utilizate și în cadrul unor tablouri cu elemente de tip structură.

Exemplu: `struct complex tablou_z[10]; // tablou unidimensional care conține 10 elemente de tip complex`

Accesul la elementele tabloului se face la fel ca la variabilele simple: `tablou_z[i].real` – partea reală a celui de-al i-lea element al tabloului.

### 3.4.5. Pointeri la structuri

Structurile pot fi parametri de funcții și pot fi returnate de către funcții. Un pointer la o structură se declară la fel ca și în cazul unei alte variabile.



`struct nume_structură *nume_pointer;`

Exemplu: `struct complex *pz;`

Accesul la elementele unei structuri folosind pointerii se face cu ajutorul operatorului '`->`'.

Exemplu: `z->real=3; z->imaginare=-2;`

Considerând p un pointer și m un membru al structurii către care pointează p, se pot folosi operatorii de incrementare și decrementare, care au următorul efect:



Expresie	Efect
<code>(++p)-&gt;m</code>	incrementează p și apoi acceseză pe m
<code>++p-&gt;m</code>	acceseză pe m și apoi incrementează p
<code>(p-&gt;m)++</code>	incrementează pe (p->m) după acces
<code>++(p-&gt;m)</code>	incrementează pe (p->m) înainte de acces

### 3.4.6. Tipuri definite de utilizator

Se pot defini tipuri utilizator care să îmbunătățească scrierea programelor.

Forma generală este următoarea:



```
typedef tip_de_date_existent tip_de_date_nou;
```

Exemplu: `typedef int INTREG;`

Tipul de date `int` l-am redenumit cu `INTREG`, astfel încât peste tot în cadrul programului respectiv, unde am făcut definirea, putem folosi noua denumire

Și pentru structuri putem redefini astfel:



Exemplu:

```
typedef struct complex
{
    double real, imaginari;
}COMPLEX;
sau
typedef struct
{
    double real, imaginari;
}COMPLEX;
```

### 3.4.7. Uniuni

În timp ce în cazul structurilor, compilatorul C alocă spațiu de memorie pentru fiecare membru al structurii și lucrează cu această structură ca un tot unitar, mai există o posibilitate ca să se aloce o singură zonă de memorie pentru toți membri structurii. Pentru aceasta se folosește tipul de date utilizator uniune - **union**.

O uniune se declară astfel:



```
union nume_uniune  
{  
    elementele uniunii  
};
```

## 3.5. Probleme rezolvate

### 3.5.1. Enunțuri

1. Să se creeze o structură care să rețină partea reală și partea imaginară a unui număr complex, iar apoi să se creeze câte o funcție pentru următoarele operații cu numere complexe: adunare; scădere; înmulțire; conjugare; calculul modulului unui număr complex.

2. Să se scrie o funcție care calculează pătratele elementelor unei matrice, fără modificarea matricei ca parametru de intrare. Funcția *main()* va inițializa matricea, va afișa conținutul matricei înainte și după efectuarea calculelor.

Indicație:

Stim că, un masiv (tablou unidimensional sau bidimensional) este transmis ca parametru de intrare într-o funcție prin adresă. Singura modalitate de transmitere prin valoare este construirea unei structuri având ca membru masivul respectiv.

3. Să se scrie o funcție care calculează pătratele elementelor unei matrice, modificând matricea inițială. Funcția *main()* va initializa matricea, va afișa conținutul matricei înainte și după efectuarea calculelor.

Indicație: Spre deosebire de problema anterioară, de această dată se dorește ca modificările să opereze în matricea inițială. Transferul prin adresă este posibil și în acest caz utilizând un pointer la structura dată.

4. Utilizând tipul de date *union*, să se scrie un program care să ofere utilizatorului mai multe opțiuni de alegere a unei figure geometrice dorite (cerc, pătrat sau dreptunghi) și apoi să-i calculeze aria.

### 3.5.2. Soluții propuse

#### Problema 1:

Să se creeze o structură care să rețină partea reală și partea imaginară a unui număr complex, iar apoi să se creeze câte o funcție pentru următoarele operații cu numere complexe: adunare; scădere; înmulțire; conjugare; calculul modulului unui număr complex.



```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>

typedef struct
{
    double re; // re = partea reală a numărului
    double im; // im = partea imaginara a numărului
} complex;

int main(void)
{
    complex x,y,z; // x,y,z sunt de tip COMPLEX
    complex adunare(complex a,complex b); // tipul returnat de
    functii
    complex scadere(complex a,complex b); // este tot COMPLEX
```

```

complex inmultire(complex a,complex b);
    complex conjugat(complex a);
    double modul(complex a);
    int optiune; // variabila folosita pentru a alege varianta
dorita
    double w;
    char raspuns; // variabila folosita pentru continuarea
introducerii
                                // datelor de intrare
    clrscr();
    do{
        cout<<" Introduceti partea reala a numarului X : "
;cin>>x.re;
        cout<<" Introduceti partea inimaginara a numarului X :
";cin>>x.im;
        cout<<" Introduceti partea reala a numarului Y :
";cin>>y.re;
        cout<<" Introduceti partea imaginara a numarului Y :
";cin>>y.im;
        cout<<" \n Doriti sa continuati ( D / N ) : "<<endl;
        raspuns=getch();
    }while(raspuns=='d' || raspuns=='D');
    clrscr();
    while(1)
    {
        cout<<" Operatia :"<<endl;
        cout<<" 1 : Adunarea a doua numere complexe\n";
        cout<<" 2 : Scaderea a doua numere complexe \n";
        cout<<" 3 : Inmultirea a doua numere complexe \n";
        cout<<" 4 : Conjugatul unui numar complex\n";
        cout<<" 5 : Modulul unui numar complex\n";
        cout<<" 6,7,8,... : EXIT\n";
        cin>>optiune;
        switch (optiune) {
            case 1 :
                z=adunare(x,y);
                cout<<"Adunare z = " <<z.re << " + " << z.re<<
"\n";
                delay(5000); // intarzie afisarea pe ecran a datelor
                clrscr();
                break;
            case 2 :
                z=scadere(x,y);
                cout<<"Scadere z = " <<z.re<< " + " <<z.im<<
"\n";
                delay(5000);
                clrscr();
                break;
        }
    }
}

```

```

        case 3 :
            z=inmultire(x,y);
            cout<<"Inmultirea lui x cu y este z = "<<z.re<<
+ "<<z.im<<" *i"<<endl;;
            delay(5000);
            clrscr();
            break;
        case 4 :
            z=conjugat(x);
            cout<<"Conjugatul lui x este z = "<<z.re<<
+ "<<z.im<<" *i"<<endl;
            delay(5000);
            clrscr();
            break;
        case 5 :
            w=modul(x);
            cout<<"Modulul lui x este w = "<<w<<endl;
            delay(5000);
            clrscr();
            break;
        default : exit(0);
    }
}
getch();
}

complex adunare(complex a,complex b)
{
    complex rezultat;
    rezultat.re=a.re+b.re;
    rezultat.im=a.im+b.im;
    return rezultat;
}

complex scadere(complex a,complex b)
{
    complex rezultat;
    rezultat.re=a.re-b.re;
    rezultat.im=a.im-b.im;
    return rezultat;
}

complex inmultire(complex a,complex b)
{
    complex rezultat;
    rezultat.re=a.re*b.re-a.im*b.im;
    rezultat.im=a.re*b.im-a.im*b.re;
    return rezultat;
}

```

```

complex conjugat(complex a)
{
    complex rezultat;
    rezultat.re=a.re;
    rezultat.im=-a.im;
    return rezultat;
}

double modul(complex a)
{
    double rez;
    rez=sqrt(a.re*a.re+a.im*a.im);
    return rez;
}

```

### Problema 2:

Să se scrie o funcție care calculează pătratele elementelor unei matrice, fără modificarea matricei ca parametru de intrare. Funcția main() va inițializa matricea, va afișa conținutul matricei înainte și după efectuarea calculelor.

Indicație:

Stim că, un masiv (tablou unidimensional sau bidimensional) este transmis ca parametru de intrare într-o funcție prin adresă. Singura modalitate de transmitere prin valoare este construirea unei structuri având ca membru masivul respectiv.



```

#include<iostream.h>
#include<conio.h>
struct matrice{
    int a[3][3];
};
void calculmat(struct matrice mat);
void printmat(struct matrice mat);
int main()
{
    struct matrice mat={1,2,3,4,5,6,7,8,9};
    clrscr();
    cout<<endl<<" Matricea inainte de calcule : "<<endl;
    printmat(mat);
    calculmat(mat);
}

```

```

cout<<endl<<"Matricea dupa revenirea din subprogram:
\n";
    printmat(mat);
    getch();
}
void calculmat(struct matrice mat)
{
    int i,j;
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) mat.a[i][j]*=mat.a[i][j];
    cout<<"\nMatricea afisata in functia de calcul : \n";
    printmat(mat);
}
void printmat(struct matrice mat)
{
    int i,j;
    for(i=0;i<3;i++)
    {
        cout<<endl;
        for(j=0;j<3;j++) cout<<mat.a[i][j]<<" ";
    }
}

```

### Problema 3:

Să se scrie o funcție care calculează pătratele elementelor unei matrice, modificând matricea inițială. Funcția *main()* va initializa matricea, va afișa conținutul matricei înainte și după efectuarea calculelor.

Indicație:

Spre deosebire de problema anterioară, de această dată se dorește ca modificările să opereze în matricea inițială. Transferul prin adresa este posibil și în acest caz utilizând un pointer la structura dată.



```

#include<iostream.h>
#include<conio.h>
struct matrice{
    int a[3][3];
};
void calculmat(struct matrice *pmat);
void printmat(struct matrice mat);

```

```

int main()
{
    struct matrice mat={1,2,3,4,5,6,7,8,9},*pmat;
    clrscr();
    pmat=&mat;
    cout<<"\n Matricea inainte de calcule : ";
    printmat(mat);
    calculmat(pmat);
    cout<<"\n Matricea dupa revenirea din subprogram : ";
    printmat(mat);
    getch();
}
void calculmat(struct matrice *pmat)
{
    int i,j;
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) pmat->a[i][j]*=pmat->a[i][j];
    cout<<"\n Matricea afisata in functia de calcul : ";
    printmat(*pmat);
}
void printmat(struct matrice mat)
{
    int i,j;
    for(i=0;i<3;i++)
    {
        cout<<endl;
        for(j=0;j<3;j++) cout<<mat.a[i][j]<<" ";
    }
}

```

#### Problema 4:

Utilizând tipul de date *union*, să se scrie un program care să ofere utilizatorului, mai multe opțiuni, de alegere a unei figure geometrice dorite (cerc, pătrat sau dreptunghi) și apoi să-i calculeze aria.



```

#include<iostream.h>
#include<stdlib.h>
#include<conio.h>
#define cerc 0
#define patrat 1
#define dreptunghi 2
#define pi 3.14159265358979

```

```

typedef struct {
    int tip;
    union {
        int raza;
        int lp;
        int ld[2];
    } fig;
} FIG;

double aria (FIG *p)
{
switch(p->tip) {
    case cerc: return pi*p->fig.raza*p->fig.raza;
    case patrat: return p->fig.lp*p->fig.lp;
    case dreptunghi: return p->fig.ld[0]*p->fig.ld[1];
    default: return 0;
}
}

int citfig(FIG *p)
{
    char t[255];
    switch(p->tip)
    {
        case cerc: { cout<<"Dati raza: ";
                      cin>>p->fig.raza;
                      return 1;
                  };
        case patrat:
        {   cout<<"Dati latura: ";
            cin>>p->fig.lp;
            return 1;
        };
        case dreptunghi:
        {   cout<<"Dati laturile: ";
            cin>>p->fig.ld[0];
            cin>>p->fig.ld[1];
            return 1;
        };
        default: return 0;
    }
}

```

```

int main(void)
{
    char lit[2];
    FIG f;
    float a;
    clrscr();
    cout<<"Tastati una din literele: C P sau D" << endl;
    cout<<"C - pentru CERC" << endl;
    cout<<"P - pentru PATRAT" << endl;
    cout<<"D - pentru DREPTUNGHI" << endl;
    cin>>lit;
    switch(lit[0])
    {
        case 'c':f.tip=cerc;break;
        case 'd':f.tip=dreptunghi;break;
        case 'p':f.tip=patrat;break;
        default:cout<<"Nu s-a tastat unul din caracterele
c, p sau d" << endl;
    };
    citfig(&f);
    a=aria(&f);
    cout<<"Aria este: "<<a;
    getch();
}

```

### 3.6. Probleme propuse spre rezolvare

1. Tocmai s-a încheiat primul tur de scrutin pentru disputatele alegeri ale primarului din orașul dumneavoastră. Să facem o analiză a voturilor primite de cei patru candidați la scaunul de primar. Dorim să aflăm câte voturi a primit fiecare dintre candidați în fiecare cartier al orașului și câte voturi totale a primit fiecare candidat în parte. Imagineați o structură de date adecvată rezolvării acestei probleme și scrieți apoi un program care o rezolvă.

2. O companie vrea să ştie procentul salariilor totale și cheltuielile totale atribuibile fiecărui angajat. Pentru fiecare angajat se introduce de la tastatură o linie de date conținând : numele persoanei (maxim 20 de caractere), urmat de o virgulă, urmată de salariul total al persoanei (întreg) și cheltuielile (real). Scrieți un program care realizează un raport cu un header conținând totalele salariilor și cheltuielile totale pe firmă. După acest header să se afișeze un tabel conținând câte o linie pentru fiecare angajat astfel: numele, prenumele, procentul salariului total, procentul cheltuielilor totale. Procentul se calculează relativ la totalul pe firmă.
3. Se citesc data nașterii unei persoane și data curentă. Să se stabilească vârsta în ani.

Exemplu : Dacă data nașterii este 30.10.1980 iar data curentă este 01.06.2002 atunci persoana are 21 de ani.

4. Se citesc două fractii  $p$  și  $q$ , sub forma (numărător, numitor). Să se afle forma ireductibilă a fracției  $p+q$ .
5. Se citesc  $n$  numere complexe sub forma (parte reală, parte imaginară). Să se afișeze ordonat crescător sirul modulelor acestor numere complexe.

*Exemplu :* Pentru  $n=5$  și numerele  $2+3i$ ,  $3-i$ ,  $7-2i$ ,  $1-i$ ,  $3i$ , se va afișa  $1.4142$ ,  $1.7320$ ,  $3.1622$ ,  $3.6055$ ,  $7.2801$ .

6. La un concurs de informatică au participat  $n$  elevi, pentru fiecare elev cunoșcându-se numele, liceul de proveniență (șiruri de caractere) și nota obținută (de la 1 la 10). Realizați un program care citește datele elevilor参入者, apoi tipărește numele elevilor cu cea mai mare notă, precum și media generală a concurenților.

*Exemplu :* Pentru n=4 și următoarele date: (Ionescu, CNET, 10), (Antonescu, CNTV, 8), (Iliescu, CNET, 9), (Lazaroiu, CNTA, 10), elevii cu cea mai mare notă sunt Ionescu și Lazaroiu iar media generală a tuturor elevilor este 9.25.

7. Definiți structurile de date necesare pentru evidența cărților existente într-o bibliotecă, știind că pentru fiecare carte se înregistrează: titlul cărții, prețul, numărul de exemplare și valoarea cărții respective (prețul \* numărul de exemplare). Scrieți apoi un program care, prin intermediu unui meniu, selectează în mod repetat, atât timp cât utilizatorul dorește acest lucru, una din următoarele acțiuni posibile: introducerea datelor aferente unei noi cărți, determinarea valorii totale a cărților existente în bibliotecă și afișarea informațiilor aferente tuturor cărților.

8. Scrieți un program care să citească temperaturile măsurate din oră în oră precum și cantitățile zilnice de precipitații dintr-o lună a anului. Apoi, programul trebuie să afișeze:

- a) temperatura maximă (împreună cu ziua și ora asociată)
- b) temperatura minimă (împreună cu ziua și ora asociată)
- c) lista zilelor, ordonată descrescător în funcție de cantitatea de precipitații
- d) media precipitațiilor zilnice din luna respectivă a anului.

9. Scrieți un program pentru evidența studentilor unei facultăți, care să permită alegerea repetată a uneia din opțiunile de mai jos, cât timp utilizatorul dorește aceasta:

- ✓ adăugarea unui student în grupă;
- ✓ listarea tuturor studentilor din grupă;
- ✓ afișarea informațiilor despre un anumit student, căutat după nume.

Despre fiecare student al unei grupe se cunosc numele studentului, media la sfârșitul unei sesiuni și valoarea bursei.

10. Se citesc de la tastatură date despre situația școlară a n studenți: cod student, grupa, nume student, sex, număr examene (m), nota<sub>1</sub>, nota<sub>2</sub>, ..., nota<sub>m</sub>. Să se listeze studenții care au examene nepromovate și, respectiv,

numărul examenelor nepromovate. Un examen este nepromovat dacă nota este sub 5.00.

11. Se consideră un tablou cu înregistrări ce conțin informații referitoare la elevii unei școli, organizate astfel: nume, clasa, număr note (n), nota<sub>1</sub>, nota<sub>2</sub>, ..., nota<sub>n</sub>. Scrieți un program care să ordeneze elevii descrescător după medii apoi stabiliți clasa cu cei mai mulți elevi care au toate notele cel puțin 7.

12. Să se descompună un număr în factori primi, memorând rezultatul sub forma unui vector de înregistrări; fiecare înregistrare va cuprinde două câmpuri, unul indicând factorul prim, iar celălalt puterea la care apare în descompunere.

*Exemplu :* Pentru n=1960 se obține vectorul ((2,3). (5,1), (7,2)) unde primul număr din fiecare paranteză reprezintă factorul prim iar al doilea reprezintă exponentul.

13. Să se scrie un program care să administreze un parc de automobile. Informațiile relative la un automobil sunt: numărul de locuri, puterea (în cai putere), marca, numărul de înmatriculare, tipul de carburant (benzina sau motorina), natura (berlină, break sau decapotabilă). Programul trebuie să permită intrarea unei mașini, ieșirea unei mașini, înlocuirea unei mașini cu alta de același model (având alt numar de înmatriculare).

14. Se consideră un vector a cuprinzând articole de tipul:

```
struct { int x,y; }
```

- Să se ordeneze vectorul a descrescător după câmpul x, iar în caz de egalitate se sortează crescător după câmpul y;
- Să se ordeneze acele componente ale lui a care au câmpul x impar, crescător după câmpul y, celelalte componente ale lui a rămânând pe poziția lor (cea de după punctul a)).
- Să se calculeze suma tuturor câmpurilor x și produsul tuturor câmpurilor y pare din a.

*Exemplu :* Pentru  $n = 10$  și  $a = ((1,5) , (2,1), (7,7), (9,5), (2,6), (12,2), (4,4), (-1,4), (3,8), (6,22))$  se afișează:

- a)  $(12,2), (9,5), (7,7), (6,22), (4,4), (3,8), (2,1), (2,6), (1,5), (-1,4);$
- b)  $(12,2), (-1,4), (9,5), (6,22), (4,4), (1,5), (2,1), (2,6), (7,7), (3,8);$
- c)  $s=45, p=5913600.$

**15.** Se dau  $n$  puncte în plan prin coordonatele lor  $(x,y)$ . Să se listeze toate punctele care se află în interiorul cercului  $c(x_0,y_0,R)$ .

*Exemplu :* Dacă  $n=10$  și punctele sunt  $(1,5) , (2,1) , (7,7) , (9,5) , (2,6) , (12,2) , (4,4) , (-1,4) , (3,8) , (6,22)$ , iar  $x_0=2$ ,  $y_0=1$ ,  $R=5$ , se vor afișa punctele  $(1,5)$ ,  $(2,1)$ ,  $(4,4)$ ,  $(-1,4)$  (se observă că punctul  $(2,6)$  se află pe conturul cercului deci nu face parte din interiorul acestuia).

**16.** Se dau  $n$  puncte în plan, prin coordonatele lor  $(x,y)$ . Să se afișeze indicii punctelor aflate pe prima bisectoare și indicii punctelor aflate pe una din axele Ox sau Oy.

*Exemplu :* Pentru  $n=7$  și punctele  $(3,7), (0,20), (3,3), (5,0), (0,10), (-5,-5), (4,9)$  punctele 3 și 6 se află pe prima bisectoare iar punctele 2, 4 și 5 se află pe axe.

**17.** Scrieți un program care simulează amestecarea unui pachet de cărți de joc (cu 52 de cărți).

**18.** Să se scrie un program pentru admiterea la un liceu care să permită:

- a) inițializarea vectorului de înregistrări de tip elev;
- b) adăugarea unui elev;
- c) eliminarea unui elev cu un nume dat;
- d) eliminarea elevului de pe o poziție dată;
- e) calcularea mediilor;
- f) listarea alfabetică a elevilor;

- g) listarea elevilor în ordine descrescătoare a mediilor;
- h) listarea tuturor elevilor cu medii peste o medie dată.

Se presupune că examenul constă din două probe.

**19.** Se citesc de la tastatură date despre materialele aflate într-o magazie:

- a) codul materialului (întreg)
- b) denumirea materialului
- c) preț
- d) stoc normal
- e) cantitate intrată
- f) cantitate ieșită.

Să se afișeze pe ecran (sub forma de tabel) o situate a materialelor cu stoc supranormativ (cantitate intrată - cantitate ieșită > stoc normat).

**20.** Se citesc de la tastatură date despre angajații unei firme. Să se afișeze pe ecran (sub formă de tabel) angajații care îndeplinesc concomitent condițiile de funcție și sex, egale cu valorile introduse de la tastatură. Datele despre angajați sunt următoarele:

- a) marca
- b) numele și prenumele
- c) codul funcției
- d) sex
- e) salariu

Corespondența dintre codul funcției și denumirea funcției este memorată într-un vector de articole. La afișare va apărea denumirea funcției în locul codului.

**21.** Să se simuleze funcționarea unui joc LOTO. La joc participă  $n$  persoane, fiecare cu câte un bilet, pe fiecare bilet se trec  $m$  numere, iar din urnă se extrag  $p$  numere cuprinse între 1 și 99. ( $m$ ,  $n$ ,  $p$  și numerele caștigătoare se introduc de la tastatură). Pentru fiecare jucător se cunosc: numele, seria biletului cumpărat (două litere și șase cifre), precum și cele  $m$  numere pe care le-a trecut pe biletul său. Să se afișeze participanți în ordinea descrescătoare

a numărului de numere "ghicite". La același punctaj are prioritate cel care a ghicit numere situate mai aproape de începutul extragerii.

**22.** Despre cei n elevi ai unei clase se cunosc numele, înălțimea și greutatea, ultimele două informații fiind numere naturale. Se cere să se creeze o listă a elevilor, ordonată descrescător în funcție de înălțime, iar la înălțimi egale în funcție de greutate. În final se cere să se calculeze, sub forma de fracție ireductibilă, media rapoartelor înălțime/greutate. Datele se citesc de la tastatură.

**23.** La o stație meteo se alcătuiește zilnic, un buletin meteo care conține:

- a) data;
- b) numele meteorologului de serviciu;
- c) presiunea atmosferică;
- d) temperatura atmosferică;

din ziua respectivă. Să se afișeze temperatura maximă atinsă în perioada urmărită, precum și zilele în care s-a înregistrat maxima, împreună cu numele meteorologilor de serviciu din zilele respective.

**24.** Pentru evidența medicamentelor dintr-un depozit, sunt necesare următoarele informații:

- a) codul medicamentului (întreg);
- b) denumire (șir de 30 de caractere);
- c) pretul medicamentului (întreg);

Presupunând că în depozit există n medicamente se cere să se majoreze prețurile tuturor medicamentelor cu p% și să se afișeze medicamentele cu noile prețuri.

$$p = \begin{cases} 5 & daca codul medicamentului \leq 1000 \\ 10 & in caz contrar \end{cases}$$

**25.** Să se construiască o structură de date pentru reprezentarea datei calendaristice și scrieți un program C++ care să efectueze următoarele:

- a) verifică dacă valoarea unei variabile din structura de date este o dată validă
- b) calculează data calendaristică care urmează unei anumite date
- c) calculează data calendaristică care precede o anumită dată.

**26.** Să se construiască o structură de date în care să se rețină informațiile despre persoane împreună cu punctajele obținute de acestea la un concurs. Să se ordoneze descrescător în funcție de punctaj și să se afișeze lista ordonată.

**27.** Să se scrie un program C++ pentru admiterea la o facultate și care să aibă următoarele facilități:

- a) inițializarea bazei de date a concursului (a tabloului de structuri de tip student);
- b) adăugarea unui student în baza de date;
- c) înlocuirea unui student cu un alt student;
- d) inserarea unui student pe o anumită poziție în baza de date;
- e) eliminarea unui student de pe o anumită poziție din baza de date;
- f) eliminarea din baza de date a unui student având un anumit nume;
- g) căutarea unui student după nume;
- h) calcularea mediilor
- i) afișarea alfabetică a studenților;
- j) afișarea studenților în ordinea descrescătoare a mediilor;
- k) afișarea tuturor studenților cu medii peste o anumită valoare dată;
- l) eliminarea ultimului student din baza de date;

**28.** Fiind date două fracții  $p$  și  $q$ , să se scrie un program C++ care să afișeze forma ireductibilă a fracțiilor  $p+q$ ,  $p-q$  și  $p/q$ , punând în evidență numărătorul și numitorul acestora.

### 3.7. Teste grilă



3.7.1. Precizați care dintre declarațiile de mai jos reprezintă corect sirul de caractere "abcde".

- a) `char c[4] = "abcde";`
- b) `char c[5] = "abcde" ;`
- c) `char c[6] = "abcda";`
- d) `char c[ ] = "abcde";`
- e) Nici una din variantele anterioare.



3.7.2. Se consideră declarația următoare:

`char c[9] = "abcde";`

Care dintre următoarele afirmații sunt adevărate ?

- a) Declarația dată este eronată.
- b) `c[5]` este '0'
- c) `c[2]` este 'b'
- d) `c[0]` este 'a'
- e) `c [7]` are valoarea 0 binar



3.7.3. Care dintre secvențele de program de mai jos afișează corect textul "C++"?

a)  
`char s[4] = "C++";  
printf("%s", s);`

b)  
`char s[3] = "C++";  
cout << s;`

c)  
`char s = "C++";  
puts(s);`

d)  
`char s [4] ;  
printf("%s", s = "C++");`

e)  
`char s [3] ;  
cout << (s = "C++");`



3.7.4. Care dintre secvențele de mai jos realizează corect citirea șirurilor s1 și s2 ? (Se presupune că variabilele s1 și s2 sunt declarate ca șiruri de caractere)

- a) gets (s1, s2);
- b) gets (s1); gets(s2);
- c) scanf("%s %s", s1, s2);
- d) scanf ("%s" ,&s1); scanf ("%s" ,&s2);
- e) cin >>s1 >>s2;



3.7.5. Care dintre instrucțiunile programului de mai jos sunt eronate ?

```
#include <stdio.h>
#include <string.h>
int main()
{
    char a[10], b[10];    int k;          // (1)
    scanf("%s %s",a,b);      // (2)
    k=strlen (a) /2 ;        // (3)
    a[k]='*';                // (4)
    printf ("%d",strlen(a)<strlen(b)); // (5)
    b=a;                      // (6)
}
```

- a) Declarațiile de variabile din linia (1)
- b) Citirea șirurilor din linia (2)
- c) Atribuirile din liniile (3) și (4)
- d) Afisarea din linia (5)
- e) Atribuirea din linia (6)



3.7.6. Se consideră programul:

```
#include <iostream.h>
#include <string.h>
int main()
{
    char s [20] ;
    int i,nr=0;
    cin>>s;
    for (i=0;i<strlen(s) ;i++)
        if (s[i]>='0' && s[i]<='9') nr++;
    cout<<nr ;
}
```

Care dintre afirmațiile de mai jos sunt adevărate?

- a) Citirea de la tastatură este eronată
- b) Dacă de la tastatură se introduce sirul "ab06x&", atunci ciclul are șase pași
- c) Dacă de la tastatură se introduce sirul "A72m#8", atunci programul afișează valoarea 3
- d) Condiția din linia if este eronată
- e) Programul afișează valoarea inițială a lui nr, adică 0, indiferent ce sir introducem de la tastatură



3.7.7. Precizați ce sir de caractere se va afișa în urma execuției programului următor:

```
#include <iostream.h>
#include <string.h>
int main()
{
    char s[20] = "BorLanD C++ 3.1";
    int i;
    for(i=0; i<strlen(s);
        if ( (s[i]>='A') && (s[i]<='Z') )
            s[i]-=(‘A’-‘a’);
    cout<<s;
}
```

- a) "BorLanD C++ 3.1"
- b) "bORIAND C++ 3.1"
- c) "BORLAND C++ 3.1"
- d) "borland C++ 3.1"
- e) "Borland C++ 3.1"



3.7.8. Se consideră un sir a definit ca un vector de caractere.

Care dintre cele două secvențe S1) și S2) de mai jos afișează prima literă mare din sir?

a) numai S1)

```
S1)
i=0;
while(a[i] && ( a[i]<'A' || a[i]>'Z' )) i++;
printf ("%c",a[i]);
```

b) numai S2)

c) nici una

d) ambele

e) sevențele conțin erori

```
S2)
i=0;
do {
    i++;
} while (a [i] && !(a[i]>='A' && a[i]<='Z'));
putchar (a[i]);
```



3.7.9. Precizați ce text va tipări programul de mai jos:

a) Popescu

b) Ionescu

c) Vasilescu

d) Nimic

e) Programul conține erori

```
#include<stdio.h>
int main()
{
    char v[3][100] = {"Popescu", "Ionescu",
    "Vasilescu"};
    if ( v[0] < v[1] ) printf("%s",v[0]);
    else printf("%s",v[1]);
}
```



3.7.10. Care dintre cele trei instrucțiuni printf ale funcției main de mai jos tipăresc succesiunea de caractere "bd" ?

```
#include <stdio.h>
int main()
{
    char s[6] [2]={ "ab", "ac", "ad", "bc" , "bd", "cd"};
    printf ("%c%c",s[3] [0],s[2] [1]);
    printf ("%s",s[3] [0]+s[2] [1]);
    printf ("%s", s [5]);
}
```

a) Toate

b) Numai prima

c) Numai primele două

d) Numai prima și a treia

e) Nici una



3.7.11. Ce va afișa programul de mai jos?

```
#include <iostream.h>
int main()
{
    char s[10] = "AB6X92P3M", b[10]; int i=0, k=0;
    while (s[i])
    {
        if (i%2) b[k]=s[i];
        i++;
    }
    b[k]=0;
    cout<<b;
}
```

- a) BX23
- b) A69PM
- c) B
- d) 3
- e) nimic



3.7.12. Fie *s* și *t* două variabile de tipul "vector de caractere". Scrieți o secvență de program pentru enunțul "dacă sirul *s* conține cel puțin *n* caractere, atunci copiază în *t* primele *n* caractere ale lui *a*".

- a) if (strlen(s)>=n) strcpy(s,t,n); t[n]=0;
- b) if (strlen(s)>=n) strcpy(t,s,n); t[n]=0;
- c) if (strlen(s)>=n) strcpy (n, s , t) ; t[n]=0;
- d) if (n<=strlen(s)) strcpy(s,t,n); t[n]='0';
- e) if (strlen(s)>=n) strcpy(t,s,n); t[n]='0';



3.7.13. Fie declarațiile: char s1[15], s2[15], s3[15];

Instrucțiunea

```
if (strcmp(s1,s2) && strcmp(s2,s3)) cout<<1;
```

va afișa valoarea 1 dacă:

- a) Sirurile *s1*, *s2* și *s3* sunt distincte două câte două
- b) Toate cele trei siruri *s1*, *s2* și *s3* sunt egale
- c) Sirurile *s1*, *s2* și *s3* sunt în ordine alfabetică crescătoare
- d) Sirurile *s1*, *s2* și *s3* sunt în ordine alfabetică descrescătoare
- e) Nici una dintre variantele anterioare



3.7.14. Considerând declarațiile:

char s[4] = "123", t[4];      int x=123, y;

Care dintre expresiile de mai jos au valoarea 0?

a) atoi(s) !=x;

b) itoa(x, t, 10) == s;

c) (y = atoi (s)) == x;

d) x == (atoi(itoa(x, t, 10)));

e) !strcmp(itoa(x, t, 10), s);



3.7.15. Ce va afișa programul următor?

- a) 0
- b) 6
- c) 89
- d) 689
- e) 6789

```
#include <stdlib.h>
#include <string.h>
#include <iostream.h>
int main ()
{
    char s[12] = "6789", t[12] = "6", u[12] = "89";
    long >> 0;
    strcat(t, u);
    if (strcmp(s, t)) x = atol(t);
    else x = atol(a);
    if (strcmp(s, u) > 0) x = atol(u);
    cout << "\n" << x;
}
```



3.7.16. Ce text afișează programul următor?

```
#include <stdio.h>
int main()
{
    char *s1 = "EXEMPLU SIMPLU ";
    char *s2 = "SIMPLU";
    printf("\n%.8s%.6s", s1, s2);
}
```

- a) "EXEMPLU"
- b) "EXEMPLU SIMPLU"
- c) "EXEMPLU SIMPLU SIMPLU"
- d) "EXEMPLUSIMPLU"
- e) "SIMPLU"



### 3.7.17. Ce afișează programul următor?

```
#include <stdio.h>
int main ()
{
    char *s="123456789", *t, *u;
    u=&s[4], s+=3, t=&s[1];
    printf( "%d%d%d", u==s, u==t, s==t) ;
}
```

- a) 000
- b) 001
- c) 010
- d) 100
- e) 111



### 3.7.18. Care dintre instrucțiuniile (1), (2),.. ..,(5) din funcția main de mai jos sunt eronate?

```
# include <stdio.h>
# include <string.h>
# include <iostream.h>
int main()
{
    char *s1, *s2, *s3;
    int x;
    s1="test";                                // (1)
    scanf ("%s",s2);                          // (2)
    s3=&s1;                                  // (3)
    cout << s1+s2;                            // (4)
    x=strlen(*s2);                           // (5)
}
```

- a) (2), (3) și (4)
- b) (2), (3), (4) și (5)
- c) (4) și (5)
- d) (3) și (5)
- e) (3), (4) și (5)



### 3.7.19. Fie programul:

```
#include <iostream.h>
int main ()
{
    char *s, *t, *u;
    int i,x;
    cin>>s;
    for (x=0, i=0; s[i]; t=&s[i], u=t+1, u[0]=t[0] ? x=1 : 0, i++);
    cout<<x;
}
```

În urma execuției programului, se va afișa valoarea 0, dacă:

- a) Toate caracterele șirului s sunt identice
- b) În șirul s există cel puțin două caractere succesive diferite
- c) În șirul s există cel mult două caractere succesive identice
- d) În șirul s există cel puțin două caractere succesive identice
- e) În șirul s nu există două caractere succesive identice



### 3.7.20. Considerăm următoarele noțiuni:

- A) vector de doi pointeri către caracter
- B) pointer către șir de două caractere

și următoarele declarații de variabile:

- I)      char \*a[2] ;
- II)     char (\*b) [2] ;

Precizați corespondențele corecte:

- a) A) cu I) și B) cu II)
- b) A) cu II); și B) cu I)
- c) Nu există corespondențe
- d) B) nu are corespondent
- e) Cele două declarații semnifică același lucru



### 3.7.21. Ce afișează programul de mai jos?

- a) 178
- b) 1AB
- c) 078
- d) OAB
- e) 067

```
#include <iostream.h>
int main ()
{
    char *s[5]={"012","345","678","9AB","CDE"};
    char *t, *u;
    int i;
    t=&s[1][0];
    cout<<(*t+5)==s[2][1]);
    u=&s[3];
    i=0;
    while (u[i]) cout<<u[i++];
}
```



### 3.7.22. Ce afișează programul de mai jos?

- a) -1
- b) 0
- c) 1
- d) 3
- e) 4

```
#include <string.h>
#include <iostream.h>
int main()
{
    char *s[10]={"10","00","10","10","01","11"};
    char *t="10"; int i=0, j=i-1;
    while (s[i])
    {
        if (!strcmp(s[i],t))    j=i;
    }
    cout<<j;
}
```



### 3.7.23. Se dă următoarele declarații:

- A. char \*a[4][6];
- B. char (\*b[4]) [6] ;
- C. char (\*c)[4][6];
- D. char ((\*d) [4]) [6] ;

și următoarele noțiuni:

N1. Vector de 4 elemente, fiecare element este un pointer către vector de 6 caractere.

N2. Pointer către matrice de caractere de 4 linii și 6 coloane

N3. Pointer către vector cu 4 elemente, fiecare fiind vector de 6 caractere.

N4. Matrice de 4 linii și 6 coloane, fiecare element este pointer către caracter.

Precizați corespondența corectă:

- a) (A,N1), (B, N2), (C, N3), (D, N4)
- b) (A, N4), (B, N1), (C, N2), (D, N3)
- c) (A, N4), (B, N1), (C, N3), (D, N2)
- d) (A, N2), (B, N3), (C, N4), (D, N1)
- e) Nu există corespondențe



3.7.24. Câte erori conține programul următor:

- a) nici una
- b) una
- c) două
- d) trei
- e) patru

```
int main ()  
{  
    char *(a[4][6]);  
    char b;  
    a [2][3]=&(b+2);  
    a [3][2]=&b+3;  
    *{4+a[2])=&b+1;  
    *a[1][3]=b+3;  
}
```

### 3.8. Răspunsuri la întrebările grilă

- 3.8.1. c) și d)
- 3.8.2. d) și e)
- 3.8.3. a)
- 3.8.4. b), c) și e)
- 3.8.5. e)
- 3.8.6. b) și c)
- 3.8.7. d)
- 3.8.8. a)
- 3.8.9. a)
- 3.8.10. b)
- 3.8.11. e)
- 3.8.12. b)
- 3.8.13. a)
- 3.8.14. a) și b)
- 3.8.15. d)
- 3.8.16. b)
- 3.8.17. c)
- 3.8.18. e)
- 3.8.19. d)
- 3.8.20. a)
- 3.8.21. b)
- 3.8.22. d)
- 3.8.23. b) și c)
- 3.8.24. b)



# Functii

## 4.1. Declarare. Apel. Prototip

**I**n limbajul C/C++, una din cele mai importante facilități o constituie folosirea funcțiilor. De fapt funcțiile reprezintă locul unde sunt scrise și executate instrucțiunile oricărui program.

**Definiție:** O *funcție* în C este o construcție independentă care conține declarații și instrucțiuni și care realizează o anumită acțiune.

Pentru a construi și folosi o funcție trebuie să cunoaștem trei elemente care sunt implicate în utilizarea funcțiilor:

1. Declararea funcției
2. Apelul funcției
3. Prototipul funcției

### 4.1.1. Declararea funcției

Forma generală a unei funcții este următoarea:



```
tip nume_funcție (lista de parametri)
{
    declarații locale
    instrucțiuni
}
```

Unde: **tip** – reprezintă tipul valorii returnate de funcție

**lista parametrii formali** – reprezintă variabilele folosite în cadrul funcției împreună cu tipul fiecărui dintre ele, dar care au un nume (denumiri) generice – formale – care nu trebuie neapărat să coincidă cu denumirile variabilelor folosite în alte funcții sau chiar în funcția principală.

**declarații locale** – reprezintă zona de declarare a variabilelor folosite doar în cadrul corpului funcției respective și care nu se pot folosi în alte funcții.

**instructiuni** – reprezintă secvența de instrucțiuni care formează funcția considerată.

Observații:

Dacă lista parametrilor este vidă, atunci reprezentarea declarării ei se face astfel: **tip nume\_funcție(void)**

#### 4.1.2. Apelul funcției

Apelul unei funcții se face astfel:



Unde  $p_{a1}, p_{a2}, \dots, p_{an}$ , reprezintă lista parametrilor actuali (reali) cu care se folosește funcția respectivă.

- Apelul unei funcții poate să apară într-o expresie numai dacă funcția returnează o valoare.
- Apelul poate să apară într-o instrucțiune de apel dacă funcția returnează sau nu o valoare.

#### 4.1.3. Prototipul funcției

Pentru a funcționa corect programul, orice funcție trebuie declarată anterior folosirii ei. Declararea este necesară dacă funcția este definită în altă parte decât în fișierul în care este apelată, sau dacă este definită în același fișier dar în urma apelării.

Prototipul unei funcții are următoarea formă generală:



**tip nume\_funcție(lista declarării parametrii);**

Unde **tip** – reprezintă tipul valorii returnate de funcție;

- dacă nu se specifică, atunci implicit (automat) este considerat de către compilatorul C ca fiind tipul **int**.
- dacă **tip** este **void** atunci funcția nu returnează nici o valoare, și deci poate acționa ca o procedură.

**lista declarării parametri** poate avea una din următoarele patru forme:

- 1) În listă apar numai tipul de date al parametrilor separați prin virgulă.
- 2) În listă apar atât tipul de date al parametrilor cât și numele lor separate prin virgulă
- 3) Lista nu este specificată
- 4) Lista este vidă – **void**

Exemplu: Prezentăm în continuare declararea unei funcții de tip double care trei parametri: unul întreg, unul de tip double și al treilea de tip caracter, în cele patru forme amintite mai sus:

- 1) double f(int, double, char);
- 2) double f(int i, double x, char c);
- 3) double f(); // nu înseamnă că funcția nu are parametri, ci doar compilatorul nu va mai face, la apel, verificarea tipului parametrilor
- 4) double g(void); // nu are parametri

Se recomandă folosirea formelor 1) și 2).

Următorul program conține prototipul și apoi definirea a două funcții care ridică la cub o valoare întreagă, respectiv o valoare reală:



```
#include <iostream.h>
int intreg_la_cub(int);
float real_la_cub(float);

int main(void)
{
    cout<<endl<<" 3 la cub este "<< intreg_la_cub(3);
    cout<<endl<<" 5.2 la cub este "<< real_la_cub(5.2);
}
int intreg_la_cub(int valoare)
{
    return (valoare*valoare*valoare);
}
float real_la_cub(float valoare)
{
    return (valoare*valoare*valoare);
}
```

#### 4.1.4. Parametri formali și actuali

În definiția funcțiilor *parametrii formali* sunt de fapt numele parametrilor care apar în construcția funcției.

Exemplu: Definim o funcție care următorii parametri formali: vârstă, salariu, și nr\_marcă, astfel:

```
void info_angajat(int vârstă, float salariu, int nr_marcă)
{
    // instrucțiunile funcției
}
```

Atunci când o funcție apelează o altă funcție, valorile transmise de funcția apelantă sunt *parametrii actuali* (sau *parametrii reali*). Astfel, dacă se apelează funcția cu valorile 34, 4500000.00 și 101 reprezintă parametri actuali în apelul funcției:

```
info_angajat(34, 4500000.00, 101);
```

Parametrii actuali pe care îi folosește o funcție pot fi valori constante sau variabile. Valoarea sau tipul variabilei trebuie să se potrivească cu parametrul formal.

Exemplu: Următoarea secvență de program ilustrează modul de folosire a variabilelor ca parametri actuali:

```
int vârsta_angajat = 34;  
float salariu_angajat = 4500000.00;  
int număr_marcă = 101;  
  
info_angajat(vârsta_angajat, salariu_angajat, număr_marcă);
```

Astfel, atunci când se apelează o funcție folosind ca variabile parametrii actuali, numele variabilelor utilizate nu au nici o legătură cu numele parametrilor formali. Compilatorul C va lua în considerare numai valorile pe care le au variabilele respective

#### 4.1.5. Variabile locale și variabile globale

Din punctul de vedere al vizibilității după domeniul de vizibilitate avem **variabile globale și variabile locale**.

**Variabilele locale** se declară în cadrul funcțiilor și se numesc *locale* deoarece numele și valorile lor sunt valabile doar în cadrul funcției respective.

Exemplu: Următoarea funcție numită var\_locale conține trei variabile locale numite x, y și z, cărora le atribuie valorile 10, 20 și 30 și apoi execută afișarea valorilor lor, iar în programul principal, compilatorul va genera cod de eroare la încercarea nereușită de a afișa aceleași valori ale variabilelor locale funcției considerate:



Exemplu:

```
#include<stdio.h>  
void var_locale(void)  
{  
    int x = 10, y = 20, z = 30;  
    printf("Variabila x = %d, variabila y = %d, variabila z =  
    %d", x,y,z);  
}  
int main(void)  
{  
    var_locale();  
    printf("Variabila x = %d, variabila y = %d, variabila z =  
    %d", x,y,z);  
}
```

Variabilele locale folosite în cadrul unei funcții, sunt recunoscute numai în funcția în cadrul căreia au fost declarate.

Exemplu: Funcția local declară trei variabile x, y și z:



```
Exemplu:  
void local(void)  
{  
    int x, y, z;  
    x = 30;  
    y = x + 10;  
    z = x + y;  
    printf("Variabila x = %d, variabila y = %d, variabila z =  
    %d", x, y, z);  
}
```

Variabilele globale sunt acele variabile care se declară înaintea oricărora declarații de funcții, iar numele, valorile și existența lor este recunoscută în întregul program.

Exemplu: Următorul program conține două funcții, una numită *var\_globale* și cea de-a doua funcție principală *main* și trei variabile x, y și z:



```
Exemplu:  
#include <stdio.h>  
int x = 10, y = 20, z = 30;  
  
void var_globale(void)  
{  
    printf("Variabila x = %d, variabila y = %d, variabila z  
    = %d", x,y,z);  
}  
int main(void)  
{  
    var_global();  
    printf("Variabila x = %d, variabila y = %d, variabila z =  
    %d", x, y, z);  
}
```

După compilarea și execuția acestui program, ambele funcții, *var\_globale* și *main*, vor afișa pe ecran valorile variabilelor globale. Se observă că declararea variabilelor s-a făcut în afara funcțiilor. Atunci când se declară variabile globale, toate funcțiile programului pot folosi și modifica-

valorile unei variabile globale prin simpla referire la numele său. Chiar dacă variabilele globale par convenabile la prima vedere, ele nu sunt recomandate de obicei. Astfel, dacă se folosesc variabile globale se observă că nu mai trebuie utilizați parametri în cadrul funcțiilor și deci numai trebuie și se înțeleagă mecanismul *apelului prin valoare* și *apelului prin referință*.

Totuși în loc să reducă numărul de erori, folosirea variabilelor globale măresc numărul lor. Deoarece sursa programului poate modifica valoarea unei variabile globale în orice loc al programului, este foarte dificil pentru un alt programator să găsească fiecare loc din program în care variabila respectivă se modifică. Astfel, alți programatori pot modifica programul dar fară ca să aibă vreun control asupra efectelor acestor modificări asupra variabilelor declarate global. Este o regulă generală ca orice modificare a unei variabile să se reflecte doar asupra funcției care le folosește. De aici este recomandabil ca orice program în C să aibă numai variabile locale și eventual doar câteva variabile globale (cât mai puține).

Prezentăm în continuare un exemplu de program care evidențiază efectele folosirii numelor de variabile identice atât în funcții, cât și în funcția principală:



Exemplu:

```
#include <stdio.h>
int a = 10, b = 20, c = 30; // variabile globale
void valoarea_lui_a(void)
{
    int a = 100;
    printf("variabila a contine %d; variabila b contine %d;
variabila c contine %d\n", a,b,c);
}
int main(void)
{
    valoarea_lui_a();
    printf("variabila a contine %d; variabila b contine %d;
variabila c contine %d\n", a,b,c);
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

*variabila a contine 100; variabila b contine 20; variabila c contine 300*

*variabila a contine 10; variabila b contine 20; variabila c contine 300*

Se observă că numele variabilei globale intră în conflict cu cel al variabilei locale și atunci compilatorul de C++, va folosi întotdeauna variabila locală. Deci, în funcție, se va afișa valoarea modificată a variabilei a și nu cea inițială, care a fost declarată global la începutul programului.

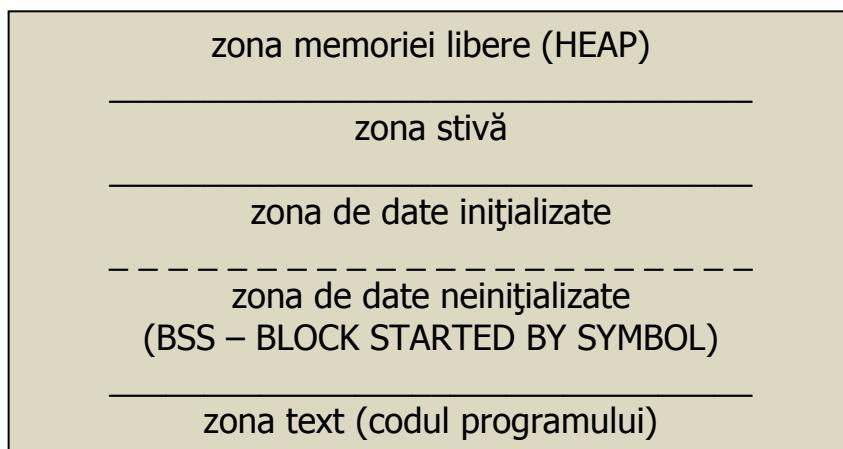
#### 4.1.6. Variabile statice și variabile automatice

Din punctul de vedere al locului în care sunt alocate variabilele și al locului alocării avem **variabile statice** și **variabile automatice (auto)**.

**Variabilele automate** se alocă în regiștri de stare sau pe stivă și sunt disponibile numai în locul în care s-a făcut alocarea (ele se alocă la execuție)

**Variabilele statice** se alocă în zona de date a programului la încărcarea programului în memorie. Ele sunt disponibile pe toată durata de existență a programului în memorie.

Prezentăm în continuare *harta simplificată a memoriei la încărcarea programului*:



Registers

Acum putem specifica pentru fiecare zonă, variabilele și parametrii care se alocă acolo:

În zona stivă se alocă:

1. variabilele locale automate
2. parametrii funcțiilor
3. adresa de return a funcției
4. variabilele temporare necesare evaluării expresiilor

În zona de date se alocă:

1. variabilele globale
2. șiruri inițializate și constante
3. variabilele locale statice

În zona registers se alocă:

1. variabilele locale automate
2. parametrii de apel ai funcțiilor

În limbajul C++, variabilele care se declară în cadrul funcției sunt adesea numite și automate, deoarece compilatorul C le creează automat când începe execuția funcției și apoi le distrugă când ea se încheie. Această caracteristică a variabilelor se explică prin faptul că variabilele funcțiilor sunt păstrate de compilator temporar în stivă. Ca urmare, funcția atribuie o valoare unei variabile în timpul unei apelări, dar variabila pierde valorile după ce funcția se încheie. La următoarea apelare a funcției, valoarea variabilei este din nou nedefinită. În funcție de procesele executate de funcția respectivă, este posibil ca variabilele funcției să memoreze ultima valoare care le-a fost atribuită în cadrul funcției.

Exemplu: Prezentăm în continuare o funcție care afișează numărul matricol pentru fiecare elev dintr-o școală. Funcția *afisează\_matricol*, folosește o variabilă statică *id\_elev* care păstrează numărul de identificare al elevului pentru care s-a tipărit ultima foaie matricolă. În acest fel, fără nici o altă mențiune funcția va începe să tipărească foaia matricolă a următorului elev:

```

void afiseaza_matricol(int numar_print)
{
    static int id_elev;
    // celealte instrucțiuni
}

```

Următorul program ilustrează folosirea unei variabile *static*, astfel încât de fiecare dată când funcția este apelată se va afișa o valoare cu o unitate mai mare decât precedenta datorită folosirii variabilei staice *id\_elev*.



```

#include<stdio.h>
void afiseaza_matricol(int numar_print)
{
    static int id_elev = 100;
    printf("Elevul cu numarul matricol %d\n", id_elev);
    id_elev++;
    // celealte instructiuni
}
int main(void)
{
    afiseaza_matricol(1);
    afiseaza_matricol(1);
    afiseaza_matricol(1);
}

```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

*Elevul cu numarul matricol 100*

*Elevul cu numarul matricol 101*

*Elevul cu numarul matricol 102*

Se observă că variabila *id\_elev* își păstrează valoarea de la o apelare la alta.

#### 4.1.7. Funcții matematice

În programele în care avem nevoie să efectuăm anumite calcule matematice, putem folosi câteva funcții predefinite ale limbajului C care efectuează aceste calcule. Prototipurile acestor funcții se află în fișierul sistem *<math.h>*, care trebuie inclus pentru compilarea programului.

Prezentăm în continuare câteva exemple de funcții matematice:

## 1. Valoarea absolută a unui număr întreg – *abs()*:



```
#include<stdio.h>
#include<math.h>
int main(void)
{
    printf("Valoarea absolută a lui %d este %d\n", 10, abs(10));
    printf("Valoarea absolută a lui %d este %d\n", 0, abs(0));
    printf("Valoarea absolută a lui %d este %d\n", -10, abs(-10));
}
```

După compilarea și execuția programului se obțin următoarele valori:

*Valoarea absolută a lui 10 este 10*

*Valoarea absolută a lui 0 este 0*

*Valoarea absolută a lui -10 este 10*

## 2. Rotunjirea unei valori reale în virgulă mobilă – *ceil()* și *floor()*:

Funcția *ceil()* se folosește atunci când se dorește să se rotunjească valoarea unei variabile sau a unei expresii, înlocuind-o cu valoarea întreagă imediat următoare.

Funcția *floor()* se folosește atunci când se dorește să se rotunjească valoarea unei variabile sau a unei expresii, înlocuind-o cu valoarea întreagă imediat anterioară.



```
#include <stdio.h>
#include <math.h>
int main(void)
{
    printf("Valoarea %f rotunjita cu functia ceil() este %f\n", 1.9,
ceil(1.9));
    printf("Valoarea %f rotunjita cu functia ceil() este %f\n", 2.1,
ceil(2.1));
    printf("Valoarea %f rotunjita cu functia floor() este %f\n", 1.9,
floor(1.9));
    printf("Valoarea %f rotunjita cu functia floor() este %f\n", 2.1,
floor(2.1));
}
```

După compilarea și execuția programului se obțin următoarele valori:

*Valoarea 1.900000 rotunjita cu funcția ceil() este 2.000000*

*Valoarea 2.100000 rotunjita cu funcția ceil() este 3.000000*

*Valoarea 1.900000 rotunjita cu funcția floor() este 1.000000*

*Valoarea 2.100000 rotunjita cu funcția floor() este 2.000000*

### 3. Funcții trigonometrice

1. *sinus – sin()*
2. *cosinus – cos()*
3. *sinusul hiperbolic – sinh()*
4. *cosinusul hiperbolic – cosh()*
5. *tangenta – tan()*
6. *tangenta hiperbolică – tanh()*
7. *arcsinus – asin()*
8. *arccosinus – acos()*
9. *arctangenta – atan()*

Într-un triunghi dreptunghic, sinusul unui unghi este raportul între latura opusă și ipotenuză. Pentru a folosi în programele C, determinarea sinusului unui unghi, se poate utiliza funcția *sin()*, care returnează o valoare de tip double ce reprezintă sinusul unui unghi specificat în radiani. La fel, se pot calcula cosinusul unui unghi și tangenta unui unghi:



```
#include <stdio.h>
#include <math.h>
int main(void)
{
    double radian;
    double pi=3.14159265;
    for(radian=0.0; radian<3.1; radian+=0.1)
        printf("Sinus de %f este %f\n",radian,sin(radian));
        printf("Cosinus de pi/2 este %6.4f\n",cos(3.14159/2.0));
        printf("Cosinus de pi este %6.4f\n",cos(3.14159));
        printf("Tangenta de pi este %f\n",tan(pi));
        printf("Tangenta de pi/4 este %f\n",tan(pi/4.0));
        printf("\n Arcsinus\n");
    for(radian=-0.5; radian<=0.5; radian+=0.2)
        printf("%f %f\n",radian,asin(radian));
```

```

printf("\n Arccosinus\n");
for(radian=-0.5; radian<=0.5; radian+=0.2)
    printf("%f %f\n",radian,acos(radian));

printf("\n Arctangenta\n");
for(radian=-0.5; radian<=0.5; radian+=0.2)
    printf("%f %f\n",radian,atan(radian));
}

```

#### 4. Funcția exponențială - $e^x$

Pentru a calcula  $e^x$  trebuie să folosim funcția `exp()`:



```

#include <stdio.h>
#include <math.h>
int main(void)
{
    double valoare;
    for(valoare=0.0; valoare<=1.0; valoare+=0.1)
        printf("exp(%f) este %f\n", valoare, exp(valoare));
}

```

După compilarea și execuția programului se obțin următoarele valori:

$\exp(0.000000)$  este 1.000000  
 $\exp(0.100000)$  este 1.105171  
 $\exp(0.200000)$  este 1.221403  
 $\exp(0.300000)$  este 1.349859  
 $\exp(0.400000)$  este 1.491825  
 $\exp(0.500000)$  este 1.648721  
 $\exp(0.600000)$  este 1.822119  
 $\exp(0.700000)$  este 2.013753  
 $\exp(0.800000)$  este 2.225541  
 $\exp(0.900000)$  este 2.459603  
 $\exp(1.000000)$  este 2.718282

#### 5. Restul împărțirii unui real la un număr real – `fmod()`:



```
#include <stdio.h>
#include <math.h>
int main(void)
{
    double numarator=10.0, numitor=3.0;
    printf("fmod(10,3) este %f\n",fmod(numarator,numitor));
}
```

După compilarea și execuția programului se obțin următoarele valori:

*fmod(10,3)=1.000000*

#### 6. Calculul părții întregi și fracționare dintr-un număr real – *modf()*:



```
#include <stdio.h>
#include <math.h>
int main(void)
{
    double valoare=1.2345;
    double parte_intreaga, fract;
    fract=modf(valoare, &parte_intreaga);
    printf("Valoarea %f are partea intreaga egala cu %f si partea
fractionara egala cu %f",valoare,parte_intreaga,fract);
}
```

După compilarea și execuția programului se obțin următoarele valori:

*Valoarea 1.234500 are partea intreaga egala cu 1.000000 si partea
fractionara egala cu 0.234500*

#### 7. Calculul lui $x^n$ – *pow()*

Ridicarea unei valori x la o putere dată n se poate efectua folosind funcția *pow()*, care utilizează o valoare de tip *double* și returnează o valoare de tip *double*:



```
#include<stdio.h>
#include<math.h>
int main(void)
{
    int putere;
    for(putere=-2; putere <= 2; putere++)
        printf("10 ridicat la puterea %d este %f\n", putere,
               pow(10.0, putere));
}
```

După compilarea și execuția programului se obțin următoarele valori:

```
10 ridicat la puterea -2 este 0.010000
10 ridicat la puterea -1 este 0.100000
10 ridicat la puterea 0 este 1.000000
10 ridicat la puterea 1 este 10.000000
10 ridicat la puterea 2 este 100.000000
```

#### 8. Calculul rădăcinii pătrate a unei valori – *sqrt()*

Pentru a calcula rădăcina pătrată dintr-o valoare de tip *double*, putem utiliza funcția *sqrt()*:



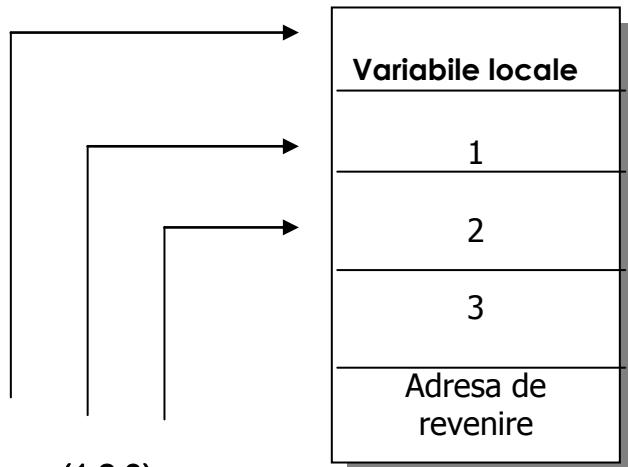
```
#include <stdio.h>
#include <math.h>
int main(void)
{
    double valoare;
    for(valoare=0.0; valoare<10.0; valoare+=0.1)
        printf("Valoarea %f are radacina patrata %f\n", valoare,
               sqrt(valoare));
}
```

#### 4.1.8. Apelul prin valoare

Înainte de a explica ce înseamnă apelul funcțiilor prin valoare, pentru a putea înțelege mai bine procesele ce au loc atât în timpul compilării, cât și în timpul execuției programelor, vom prezenta câteva aspecte despre *stiva* folosită de funcții.

Stiva este o zonă folosită pentru apelarea funcțiilor. Astfel, când programul apelează o funcție, compilatorul de C++ plasează în stivă adresa instrucțiunii care urmează apelării funcției (denumită *adresa de revenire*). Apoi, compilatorul de C plasează în stivă parametrii funcției, de la dreapta la stânga. În final, dacă funcția declară variabile locale, compilatorul alocă în stivă un spațiu pe care funcția îl utilizează pentru a stoca valoarea variabilelor

respective. În figura următoare se arată cum utilizează compilatorul de C++ stiva în cazul unui apel de funcție.



`o_funcție_oarecare(1,2,3);`

Când se termină execuția funcției, compilatorul descarcă zona de stivă care conține variabilele locale și parametrii și apoi folosește valoarea de revenire pentru a relua execuția programului de la locația corectă.

Deși utilizarea funcției este puternică, pentru că permite compilatorului să apeleze funcțiile și să le transmită informații, ea consumă timp de procesare. Intervalul de timp necesar calculatorului pentru a încărca și a descărca informațiile din stivă este numit de programatori – suprasarcina funcției (overhead). Pentru a înțelege mai bine impactul suprasarcinii funcției asupra performanțelor unui program, următorul program folosește mai întâi un ciclu repetitiv pentru a aduna valorile de la 1 la 1000000000. Apoi, utilizează un nou ciclu repetitiv pentru a aduna tot aceleași valori, dar utilizând o funcție:



```
#include <stdio.h>
#include <time.h>
float aduna(long int x, float y)
{
    float rezultat;
    rezultat = x + y;
    return(rezultat);
}
int main(void)
{
    long int i;
    float rezultat;
    time_t start_time, stop_time;
```

```

printf("lucreaza . . . \n");
time(&start_time);
for(i=1; i<=1000000000L; i++)
    rezultat+=i;
time(&stop_time);
printf("utilizare bucla %d secunde\n", stop_time - start_time);
printf("lucreaza . . . \n");
time(&start_time);
for(i=1; i<=1000000000L; i++)
    rezultat = aduna(i,rezultat);
time(&stop_time);
printf("utilizare functie %d secunde\n", stop_time - start_time);
}

```

După compilarea și execuția programului se obțin următoarele valori:

*lucreaza . . . utilizare bucla 171 secunde*

*lucreaza . . . utilizare functie 492 secunde*

Stim că programele folosesc parametri pentru a transmite informații către funcții. Atunci când se transmite un parametru unei funcții, compilatorul folosește o tehnică cunoscută sub numele de *apel prin valoare*, pentru a da funcției o copie a valorii parametrului cu care funcția va efectua prelucrările dorite. Folosind apelul prin valoare, orice modificare a parametrului funcției va exista numai în interiorul funcției apelate. La sfârșitul execuției funcției, valoarea variabilelor care i-au fost transmise rămâne nemodificată în funcția apelantă.

Exemplu: Prezentăm în continuare o funcție care prin intermediul a trei parametri x, y și z, afișează valorile lor și apoi le modifică și le afișează din nou:



```

#include <stdio.h>
void aduna(int copie_x, int copie_y, int copie_z)
{
    printf("valorile originale ale functiei sunt %d %d %d\n",
    copie_x, copie_y, copie_z);
    copie_x+=100;
    copie_y+=100;
    copie_z+=100;
    printf("valorile finale ale functiei sunt %d %d %d\n", copie_x,
    copie_y, copie_z);
}

```

```

int main(void)
{
    int x=1,y=2,z=3;
    aduna(x,y,z);
    printf("valorile finale in functia main sunt %d %d %d\n",x,y,z);
}

```

După compilarea și execuția programului se obțin următoarele valori:

*valorile originale ale functiei sunt 1 2 3*

*valorile finale ale functiei sunt 101 102 103*

*valorile finale in functia main sunt 1 2 3*

#### 4.1.9. Apelul prin referință

Sunt situații când funcțiile pe care le construim în programe trebuie să modifice valorile variabilelor transmise ca parametri. Acest lucru nu se poate realiza folosind *apelul prin valoare*, deoarece în acest caz funcția respectivă acționează asupra unor copii ale variabilelor respective și nu asupra variabilelor propriu-zise. Astfel se poate propune ca soluționarea acestui aspect să se facă cu ajutorul *apelului prin referință*. Acest tip de apel al parametrilor unei funcții presupune folosirea, în principal, a pointerilor. Astfel se transmit parametri funcțiilor folosind *operatorul de redirectare* ('\*'), iar la apelul din funcția principală se folosește *operatorul de adresă* ('&').

Prezentăm în continuare, câteva exemple de funcții care folosesc apelul prin referință:

Exemplu: Următoarea funcție folosește pointeri și adrese pentru a afișa și a modifica parametrii transmiși.



```

#include <stdio.h>
void modifica(int *x, int *y, int *z)
{
    printf("valorile originale ale functiei sunt %d %d %d\n",*x,*y,*z);
    *x+=10;
    *y+=10;
    *z+=10;
    printf("valorile finale ale functiei sunt %d %d %d\n",*x,*y,*z);
}

```

```

int main(void)
{
    int x=1,y=2,z=3;
    modifica(&x,&y,&z);
    printf("valorile finale in functia main sunt %d %d %d\n",x,y,z);
}

```

După compilarea și execuția programului se obțin următoarele valori:

**valorile originale ale functiei sunt 1 2 3**

**valorile finale ale functiei sunt 11 12 13**

**valorile finale in functia main sunt 11 12 13**

Exemplu: Următoarea funcție folosește pointeri și adrese pentru a afișa și a modifica doar un parametru transmis prin referință, iar celălalt, transmis prin valoare nu va fi modificat.



```

#include <stdio.h>
void modifica_unul(int *x, int y)
{
    *x+=20;
    y+=100;
}
int main(void)
{
    int x = 5, y = 10;
    modifica_unul(&x,y);
    printf("valorile finale in functia main sunt %d %d\n", x, y);
}

```

După compilarea și execuția programului se obțin următoarele valori:

**valorile finale in functia main sunt 25 10**

Prezentăm în continuare două programe care folosesc apelul prin valoare și două programe care folosesc apelul prin referință:

#### Problema 1:

Se consideră două numere întregi a și b. Se cere să se verifice dacă sunt sau nu *numere prime gemene*. Spunem că două numere se numesc *prime gemene* dacă sunt prime amândouă și diferența lor este egală cu 2.



```
#include <stdio.h>
#include <math.h>
int prim(long int x)
{
    long int i,nr_prim=1;
    for(i=2;i<=x/2;i++)
        if(x%i==0) nr_prim=0;
    if(nr_prim==1) return 1;
    else return 0;
}

int main(void)
{
    long int a,b;
    printf("Dati primul numar a= ");scanf("%ld",&a);
    printf("Dati al doilea numar b= ");scanf("%ld",&b);
    if( (prim(a) == 1) && (prim(b) == 1) && (abs(a-b) == 2) )
        printf("numerele %ld si %ld sunt prime gemene\n", a, b);
    else
        printf("numerele %ld si %ld NU sunt prime gemene\n", a,
b);
}
```

### Problema 2:

Se consideră două numere întregi  $a$  și  $b$ . Se cere să se calculeze și să se afișeze cel mai mare divizor comun al lor împreună cu cel mai mic multiplu comun al lor. Pentru rezolvarea primei cerințe vom folosi **algoritmul lui EUCLID**, prin **împărțiri repetate**, iar pentru cea de-a doua cerință vom folosi relația  $cmmmc(a, b) = (a * b) / cmmdc(a, b)$ .



```
#include <stdio.h>
#include <math.h>
long int cmmdc(long int x, long int y)
{
    long int r;
    r=x%y;
    while(r!=0) {      x=y;  y=r;  r=x%y;      }
    return y;
}
```

```

int main(void)
{
    long int a,b,c,d;
    printf("Dati primul numar a= ");      scanf("%ld",&a);
    printf("Dati al doilea numar b= ");    scanf("%ld",&b);
    c=cmmdc(a,b);
    d=(a*b)/c;
    printf("C.m.m.d.c. al numerelor %ld si %ld este %ld\n",a,b,c);
    printf("C.m.m.m.c. al numerelor %ld si %ld este %ld\n",a,b,d);
}

```

### Problema 3:

Se consideră un număr întreg  $a$  și se cere să se determine dacă este sau nu *palindrom*. Spunem că un număr este *palindrom* dacă este egal cu răsturnatul său (adică numărul cu cifrele de la dreapta la stânga). Pentru rezolvare construim o funcție care primește un parametru prin valoare (numărul dat) și un parametru prin referință (răsturnatul său).



```

#include <stdio.h>
int palindrom(long int x,*long int y)
{
    long int z=x;
    *y=0;
    while(x!=0){ *y=*y*10+x%10; x=x/10; }
    if(z==*y) return 1;
    else return 0;
}

int main(void)
{
    long int a,b;
    printf("Dati numarul a= ");scanf("%ld",&a);
    if(palindrom(a,&b)==1)
        printf("numarul %d este palindrom\n",a);
    else
        printf("numarul %d NU este palindrom\n",a);
}

```

### Problema 4:

Se consideră un număr întreg  $y$ . Se cere să se scrie cifrele sale într-un tablou unidimensional și apoi să se afișeze elementele tabloului astfel încât cifrele să apară în ordinea din numărul considerat. Pentru rezolvare construim

o funcție care are un parametru dat prin valoare (numărul dat), și doi parametrii dați prin referință (tabloul de cifre și dimensiunea lui).



```
#include <stdio.h>
#include <math.h>
long int y;
int a[30],n,i;
void cifre(long int x, int *a, int *n)
{
    int i=0;
    while(x!=0) { a[i]=x%10; i++; x=x/10; }
    *n=i-1;
}

int main(void)
{
    printf("Dati numarul y= ");      scanf("%ld",&y);
    cifre(y,a,&n);
    printf("Cifrele numarului sunt: ");
    for(i=n;i>=0;i--) printf("%d ",a[i]);
}
```

#### 4.1.10. Pointeri către funcții

În limbajul C++ numele unei funcții nu este o variabilă ci este un pointer constant, mai exact un pointer către zona text a programului, la fel ca la tablouri. Pointerii pot fi returnați de către funcții.

Forma generală de declarare a pointerilor la un tip de funcție este:



**tip (\*nume\_pointer) (lista de parametri formali);**

Observație: Primele paranteze sunt obligatorii deoarece expresia **tip \*nume\_pointer(lista de parametri formali)** este o funcție care returnează un pointer. Ca regulă practică, declarația unui pointer la o funcție se face la fel ca

și cum am declara o funcție obișnuită doar că în loc de numele funcției se pune expresia (**\* nume\_pointer**).

Exemplu:    char (\*p)(char \*, char \*);

Apelul unei funcții prin intermediul unui pointer se poate face în două moduri:

- a) Dacă funcția este fără tip:



**(\*nume\_pointer) (lista de parametri actuali);**

- b) Dacă funcția este cu tip:



**var=(\*nume\_pointer) (lista de parametri actuali);**

Observație: Variabila **var** trebuie să fie de tip pointer.

Exemplu: Prezentăm în continuare un program care definește un tablou unidimensional cu 6 pointeri la o funcție de tipul *double*, care un parametru de același tip și initializează acest tablou cu adresele unor funcții matematice din fișierul sistem *<math.h>*.



Observație:  
Următoarele expresii au semnificațiile:

**tablou[i]** – adresa funcției aflată pe poziția i  
**\*tablou[i]** – funcția aflată pe poziția i  
**(\*tablou[i])(x)** – apelul funcției cu argumentul x



```
#include <stdio.h>
#include <math.h>
#include <conio.h>

double(*tablou[]){sin,cos,tan,exp,log,log10};

int main(void)
{
    int i;
    double x;
    clrscr();
    for(x=0;x<=1;x+=0.01)
    {
        for(i=0;i<sizeof(tablou);i++)
            printf("%lf ",(*tablou[i])(x));
        printf("\n");
    }
}
```

#### 4.1.11. Argumentele liniei de comandă

O aplicație importantă a tablourilor de pointeri este transmiterea argumentelor către funcția *main()*.

Linia de comandă este zona de introducere și de editare a comenzi și formatul general este:

*C:\>nume\_program argument\_1 argument\_2 . . . argument\_n*

Când se lansează un program executabil de la tastatură (de exemplu: în MS-DOS, un program cu extensia .COM sau .EXE) se pot introduce și argumentele sau parametrii în linia de comandă. (de exemplu: C:\dir /p).

În limbajul C putem face ca funcția *main()* să aibă acces la aceste argumente. Astfel încât putem afișa forma generală a funcției *main()*:



```
int main(int argc, char *argv[ ]);
```

Unde: **argc** (*argument count*) – reprezintă numărul de argumente din linia de comandă inclusiv numele programului;

**\*argv[ ]** (*argument values*) – este un tablou de pointeri către caractere la argumentele liniei de comandă.

Fiecare element al tabloului se referă la argumentele liniei de comandă astfel:

*argv[0] – nume\_program*

*argv[1] – argument\_1*

*argv[2] – argument\_2*

.....

*argv[n] – argument\_n*

Exemplu: Să presupunem că avem un program în limbajul C, numit PRODUS.C cu două argumente, care înmulțește primul argument cu cel de-al doilea argument.

*C:\Probleme\produs 25 4*

Avem, în memorie, un tablou de pointeri:

`argv[ ] = {"C:\probleme\produs.exe", "25", "4", NULL }`

Exemplu: prezentăm în continuare un program care tipărește toate argumentele – inclusiv numele programului.



<b>Valoarea i</b>	<b>Continutul lui argv</b>
0	numele programului (inclusiv calea)
1	argument 1
2	argument 2
...	....
argc	argument argc
	NULL

## 4.2. Probleme rezolvate

### 4.2.1. Enunțuri

1. Să se scrie câte o funcție care să determine:

- cel mai mare divizor comun ( notat prin *cmmdc* ) a două numere întregi date ca parametri.
- cel mai mic multiplu comun ( notat prin *cmmmc* ) a două numere întregi date ca parametri

Indicații:

- ✓ Pentru determinarea cmmdc se va folosi algoritmul lui Euclid prin împărțiri repetitive.
- ✓ Pentru determinarea cmmmc se va folosi relația dintre cmmmc și cmmdc:

$$\text{cmmmc}(a,b) = (a*b) / \text{cmmdc}(a,b)$$

2. Să se scrie o funcție care să rezolve o ecuație de gradul al II-lea, unde coeficienții ecuației sunt dați ca parametri.

Funcția principală va citi coeficienții a două ecuații și va afișa soluțiile pentru fiecare ecuație în parte.

3. Să se scrie o funcție care să calculeze și să afișeze urma unei matrice pătratice date ca parametru.

*Urma unei matrice patratice* este suma elementelor aflate pe diagonala principală.

4. Să se scrie o funcție care să verifice dacă un număr întreg este sau nu palindrom (este egal cu răsturnatul său).

5. Să se scrie o funcție care să verifice dacă două numere întregi sunt *prime gemene* (sunt prime și diferența în modul este egală cu 2).

6. Să se scrie o funcție care să afișeze suma cifrelor unui număr întreg dat ca parametru.

7. Să se scrie câte o funcție fără tip care să efectueze:

- a) Citirea unei matrice pătratice

- b) Suma elementelor din matrice aflate pe diagonale principală și produsul celorlalte elemente
  - c) Afisarea elementelor din matrice
  - d) Afisarea sumei și produsului
8. Se consideră un vector  $x$  cu  $n$  numere întregi. Să se verifice dacă  $x$  reprezintă o mulțime. Se vor construi:
- a) funcție care să citească vectorul de la tastatură
  - b) o funcție care să afișeze vectorul
  - c) și a treia funcție care să tipărească mesajul "Vectorul dat reprezintă o mulțime" sau "Vectorul dat NU reprezintă o mulțime", în caz contrar.
9. Se consideră un număr natural în baza 10. Se cere să se scrie câte o funcție care să afișeze reprezentarea numărului dat în bazele 2 și 8.

Exemplu: Pentru  $n=25$

reprezentarea numărului în baza 2 este 11001  
reprezentarea numărului în baza 8 este 31

10. Se consideră un număr natural  $n$ . Se cere să se scrie câte o funcție, pentru următoarele cerințe:

- a) Să se afișeze toate numerele prime mai mici sau egale cu  $n$
- b) Să se afișeze toate numerele perfecte mai mici sau egale cu  $n$
- c) Să se afișeze toate pătratele perfecte mai mici sau egale cu  $n$ .

#### 4.2.2. Soluții propuse

##### Problema 1:

1. Să se scrie câte o funcție care să determine:
  - cel mai mare divizor comun (notat prin  $cmmdc$ ) a două numere întregi date ca parametri.
  - cel mai mic multiplu comun (notat prin  $cmmmc$ ) a două numere întregi date ca parametri

Indicații:

- ✓ Pentru determinarea cmmdc se va folosi algoritmul lui Euclid prin împărțiri repetate.
- ✓ Pentru determinarea cmmmc se va folosi relația dintre cmmmc și cmmdc:

$$\text{cmmmc}(a,b) = (a*b) / \text{cmmdc}(a,b)$$



```
#include <iostream.h>
int cmmdc(int a, int b)
{
    int r;
    r=a%b;
    while(r!=0)
    {
        a=b;
        b=r;
        r=a%b;
    }
    return b;
}
int cmmmc(int a, int b)
{
    return(a*b/cmmdc(a,b));
}
int main(void)
{
    int x,y,divizor,multiplu;
    cout<<"Dati primul numar "; cin>>x;
    cout<<"Dati al doilea numar "; cin>>y;
    divisor = cmmdc(x,y);
    cout<<"C.m.m.d.c este "<<divizor<<endl;
    multiplu=cmmmc(x,y);
    cout<<"C.m.m.m.c este "<<multiplu<<endl;
}
```

### Problema 2:

Să se scrie o funcție care să rezolve o ecuație de gradul al II-lea, unde coeficienții ecuației sunt dați ca parametri.

Funcția principală va citi coeficienții a două ecuații și va afișa soluțiile pentru fiecare ecuație în parte.



```
#include <iostream.h>
#include <math.h>
void ecuatie(float a, float b, float c)
{
    float delta,x1,x2;
    delta=b*b-4*c;
    if(delta<0) cout<<"ecuatia are solutii complexe";
    else
        if(delta==0)
        {
            cout<<"ecuatia are doua radacini egale"<<endl;
            x1=x2=-b/(2*a);
            cout<<"x1= "<<x1<<endl;
            cout<<"x2= "<<x2<<endl;
        }
        else
        {
            x1=(-b+sqrt(delta))/(2*a);
            x2=(-b-sqrt(delta))/(2*a);
            cout<<"ecuatia are urmatoarele radacini: "<<endl;
            cout<<"x1= "<<x1<<endl;
            cout<<"x2= "<<x2<<endl;
        }
    return;
}
int main(void)
{
    int a1,b1,c1,a2,b2,c2;
    cout<<"Dati coeficientii primei ecuatii "<<endl;
    cout<<"Dati coeficientul a1 ";    cin>>a1;
    cout<<"Dati coeficientul b1 ";    cin>>b1;
    cout<<"Dati coeficientul c1 ";    cin>>c1;
    ecuatie(a1,b1,c1);
    cout<<endl;
    cout<<"Dati coeficientii celei de-a doua ecuatii "<<endl;
    cout<<"Dati coeficientul a2 ";    cin>>a2;
    cout<<"Dati coeficientul b2 ";    cin>>b2;
    cout<<"Dati coeficientul c2 ";    cin>>c2;
    ecuatie(a2,b2,c2);
}
```

### Problema 3:

Să se scrie o funcție care să calculeze și să afișeze urma unei matrice pătratice date ca parametru.

*Urma unei matrice patratice* este suma elementelor aflate pe diagonala principală.



```
#include <iostream.h>
#include <math.h>
int urma(int a[10][10],int n)
{
    int suma=0,i,j;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            if(i==j) suma+=a[i][j];
    return suma;
}
int main(void)
{
    int a[10][10],i,j,n,u;
    cout<<"Dati numarul de linii si coloane n = "; cin>>n;
    cout<<"Dati elementele matricei "<<endl;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            cin>>a[i][j];
    cout<<"Matricea data este: "<<endl;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++) cout<<a[i][j]<<" ";
        cout<<endl;
    }
    u=urma(a,n);
    cout<<"Urma matricei este "<<u;
}
```

### Problema 4:

Să se scrie o funcție care să verifice dacă un număr întreg este sau nu palindrom (este egal cu răsturnatul său).



```
#include <iostream.h>
int palindrom(long int x)
{
    long int z=0,y;
    y=x;
    while(x!=0){ z=z*10+x%10; x=x/10; }
    if(z==y) return 1;
    else return 0;
}

int main(void)
{
    long int a,b;
    cout<<"Dati numarul a=";      cin>>a;
    if(palindrom(a) == 1)
        cout<<"numarul "<<a<<" este palindrom"<<endl;
    else
        cout<<"numarul "<<a<<" NU este palindrom"<<endl;
}
```

### Problema 5:

Să se scrie o funcție care să verifice dacă două numere întregi sunt **prime gemene** (sunt prime și diferența în modul este egală cu 2).



```
#include <iostream.h>
#include <math.h>

int prim(long int x)
{
    long int i,nr_prim=1;
    for(i=2;i<=x/2;i++)
        if(x%i==0) nr_prim=0;
    if(nr_prim==1) return 1;
    else return 0;
}
int main(void)
{
    long int a,b;
    cout<<"Dati primul numar a=";      cin>>a;
    cout<<"Dati al doilea numar b=";      cin>>b;
    if( (prim(a) == 1) && (prim(b) == 1) && (abs(a-b) == 2) )
        cout<<"numerele "<<a<<" si "<<b<<" sunt prime
        gemene"<<endl;
    else cout<<"numerele "<<a<<" si "<<b<<" NU sunt prime
        gemene"<<endl;
}
```

### Problema 6:

Să se scrie o funcție care să afișeze suma cifrelor unui număr întreg dat ca parametru.



```
#include <iostream.h>
long int n;
void cifre(long int n)
{
    int s=0;
    while(n!=0) {      s+=n%10;  n=n/10; }
    cout<<"Suma cifrelor este "<<s<<endl;;
    return;
}

int main(void)
{
    cout<<"Dati numarul n= ";cin>>n;
    cifre(n);
}
```

### Problema 7:

Să se scrie câte o funcție fără tip care să efectueze:

- a) Citirea unei matrice pătratice
- b) Suma elementelor din matrice aflate pe diagonale principală și produsul celorlalte elemente
- c) Afişarea elementelor din matrice
- d) Afişarea sumei și produsului



```
#include <iostream.h>
int a[10][10],n,m,s,p;
void citire(int a[10][10],int n)
{
    int i,j;
    cout<<"dati elem. matricii \n";
    for(i=1; i<=n; i++)
        for(j=1;j<=n;j++)
            cin>>a[i][j];
    return;
}
```

```

void suma(int a[10][10],int *s,int *p)
{
    int i,j;
    *s=0;*p=1;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            if(i==j) *s=*s+a[i][j];
            else *p=*p * a[i][j];
    return;
}

void afisare(int s, int p)
{
    cout<<"suma elementelor de pe diagonala principala este
"<<s<<endl;
    cout<<"produsul elementelor este "<<p<<endl;
    return;
}

void afisare_matrice(int a[10][10],int n)
{
    int i,j;
    cout<<"matricea este "<<endl;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++) cout<<a[i][j]<<" ";
        cout<<endl;
    }
    return;
}

int main(void)
{
    cout<<"Dati numarul linii si de coloane n = ";
    cin>>n;
    citire(a,n);
    afisare_matrice(a,n);
    suma(a,&s,&p);
    afisare(s,p);
}

```

### Problema 8:

Se consideră un vector  $x$  cu  $n$  numere întregi. Să se verifice dacă  $x$  reprezintă o mulțime. Se vor construi:

- funcție care să citească vectorul de la tastatură
- o funcție care să afișeze vectorul

- c) și a treia funcție care să tipărească mesajul “Vectorul dat reprezintă o mulțime” sau “Vectorul dat NU reprezintă o mulțime”, în caz contrar.



```
#include <iostream.h>
int x[100], n;
void citire(int x[100],int n)
{
    int i;
    cout<<"Dati elem. vectorului "<<endl;
    for(i=1; i<=n; i++)
        cin>>x[i];
    return;
}
void multime(int x[100],int n)
{
    int i, j, verific=1;
    for(i=1; i<n; i++)
        for(j=i+1; j<=n; j++)
            if(x[i] == x[j]) verific = 0;
    if(verific == 1) cout<<"Vectorul reprezinta o multime";
    else cout<<"vectorul NU reprezinta o multime";
    return;
}
void afisare(int x[100],int n)
{
    int i;
    cout<<"Vectorul este "<<endl;
    for(i=1; i<=n; i++)
        cout<<x[i]<<" ";
    cout<<endl;
    return;
}
int main(void)
{
    cout<<"Dati n = ";      cin>>n;
    citire(x,n);
    afisare(x,n);
    multime(x,n);
}
```

### Problema 9:

Se consideră un număr natural în baza 10. Se cere să se scrie câte o funcție care să afișeze reprezentarea numărului dat în bazele 2 și 8.

Exemplu: Pentru n=25

reprezentarea numărului în baza 2 este 11001

reprezentarea numărului în baza 8 este 31



```
#include <iostream.h>
int n;
void baza_2(int n)
{
    int i=1, j, x[30], cifre;
    while(n!=0)
    {
        x[i] = n % 2;
        n = n / 2;
        i++;
    }
    cifre = i-1;
    cout<<"Reprezentarea numarului in baza 2 este: ";
    for(i=cifre; i>=1; i--)          cout<<x[i];
    cout<<endl;
    return;
}
void baza_8(int n)
{
    int i=1,j,x[30],cifre;
    while( n != 0)
    {
        x[i] = n % 8;
        n = n /8;
        i++;
    }
    cifre = i-1;
    cout<<"Reprezentarea numarului in baza 8 este: ";
    for(i=cifre; i>=1; i--)          cout<<x[i];
    cout<<endl;
    return;
}
int main(void)
{
    cout<<"Dati n = ";      cin>>n;
    baza_2(n);
    baza_8(n);
}
```

### Problema 10:

Se consideră un număr natural n. Se cere să se scrie câte o funcție, pentru următoarele cerințe:

- a) Să se afișeze toate numerele prime mai mici sau egale cu n
- b) Să se afișeze toate numerele perfecte mai mici sau egale cu n
- c) Să se afișeze toate pătratele perfecte mai mici sau egale cu n.



```
#include <iostream.h>
#include <math.h>
int n;
void numere_prime(int n)
{
    int i,j,prim;
    cout<<"Numerele prime mai mici sau egale decat
"<<n<<" sunt "<<endl;
    for(i=2;i<=n;i++)
    {
        prim=1;
        for(j=2;j<=i/2;j++)
            if(i%j==0) prim=0;
        if(prim==1) cout<<i<<" ";
    }
    return;
}
void numere_perfecte(int n)
{
    int i, j, suma;
    cout<<"Numerele perfecte mai mici sau egale decat "<<n<<
    sunt "<<endl;
    for(i=6;i<=n;i++)
    {
        suma=0;
        for(j=1;j<=i/2;j++)
            if(i%j==0) suma=suma+j;
        if(suma==i) cout<<i<<" ";
    }
    return;
}
void numere_patrate_perfecte(int n)
{ int i, j;
    cout<<"Numerele patrate perfecte mai mici sau egale decat
"<<n<<" sunt "<<endl;
    for(i=1; i<=n; i++)
        if(sqrt(i) == floor(sqrt(i)) ) cout<<i<<" ";
    return;
}
```

```
int main(void)
{
    cout<<"Dati n = ";
    cin>>n;
    numere_prime(n);
    cout<<endl;
    numere_perfecte(n);
    cout<<endl;
    numere_patrate_perfecte(n);
}
```

### 4.3. Probleme propuse spre rezolvare

1. Realizați o funcție care să contorizeze numărul de vocale, respectiv consoane dintr-un sir.
2. Realizați o funcție care să calculeze numărul de valori de 1 ale reprezentării unui număr dat în baza 2. Inițial numărul se citește de la tastatură, în baza 10 și apoi se transformă în baza 2.
3. Numim *platou de lungime k* într-un sir de numere întregi, o secvență de k elemente identice (cu aceeași valoare). Scrieți un program care afișează toate platourile de lungime maximă existente într-un sir dat cu n elemente.

*Exemplu :* În sirul (1, 2, 2, 2, 3, 4, 4, 5, 5, 5) întâlnim platourile (2, 2, 2), (4, 4) și (5, 5, 5), iar cele de lungime maximă sunt primul și al treilea.

4. Se la tastatură un vector cu n elemente numere întregi. Să se tipărească toate perechile de elemente (nu neapărat consecutive) cu proprietatea că cel mai mare divizor comun al elementelor perechii este o valoare dată de la tastatură numită d.

*Exemplu:* Dacă vectorul este  $v = \{2, 3, 6, 5, 8\}$  cu  $n=5$  și  $d=2$ , atunci perechile căutate sunt : (2, 6), (2, 8) și (6,8).

5. Se dau două mulțimi cu câte n elemente fiecare, definite ca tablouri unidimensionale. Să se afișeze intersecția celor două mulțimi (elementele lor comune). Programul C++ va conține câte o funcție de tip *void* pentru citirea, afișarea și calculul intersecției cerute.

6. Un număr natural are *aspect de munte*, dacă cifrele sale sunt în ordine crescătoare până la o anumită poziție k și în ordine descrescătoare începând cu acea poziție.

*Exemplu:* 235754. Se citește de la tastatură un sir de numere întregi până când se introduce de două ori consecutiv aceeași valoare. Să se afișeze câte numere cu aspect de munte există în sir.

7. Se citește o matrice A cu n linii și m coloane. Se cere să se eliminate o coloană c și o linie l, valori citite de la tastatură. Programul C++ va conține câte o funcție de tip *void* pentru citirea, afișarea și eliminarea liniei și coloanei cerute.
8. Se citește o valoare naturală n, iar apoi n perechi de numere întregi. Se cere să se scrie un program C++, care să afișeze acele perechi care au proprietatea ca suma elementelor perechii este egală cu o valoare dată d.
9. Scrieți o funcție care să compare două siruri de caractere terminate cu caracterul punct.
10. Scrieți o funcție care primește ca parametru un sir de caractere și al cărui rezultat spune dacă se poate considera că sirul respectiv reprezintă un cuvânt (format doar din litere și, eventual cratimă ('-')).
11. Să se scrie o funcție care să convertească, dacă este posibil, un sir de caractere primit ca parametru într-un valoare numerică de tip real (float).
12. Să se scrie o funcție care să convertească o valoare numerică întreagă (int) primită ca parametru în sirul de caractere corespunzător acesteia.
13. Scrieți o funcție care, primind ca parametri două siruri de caractere, furnizează rezultatul comparării acestora. Se va folosi un raționament asemănător celui necesar pentru înscrierea/cautarea cuvintelor într-un dicționar.

14. Scrieți un program care să citească de la tastatură o linie de text și apoi să o afișeze “subliniind-o”, adică scriind câte o liniuță sub fiecare caracter (pe linia următoare, evident).
15. Să se scrie o funcție care, folosind ca parametri două siruri de caractere, notate **a** și **b**, arată dacă **a** reprezintă sau nu prefixul lui **b**.
16. Să se scrie o funcție care să realizeze extragerea unei “bucăți” dintr-un sir de caractere într-un sir nou, separat. Funcția va folosi ca informații cunoscute (parametri) sirul “sursă”, poziția în sirul “sursă” a primului caracter ce urmează a fi extras și numărul de caractere ce trebuie extrase.
17. Să se facă o statistică a cuvintelor de 1, 2, ..., 10 litere dintr-un text. Cuvintele sunt separate între ele prin unul sau mai multe spații.
18. Într-un text citit de la tastatură să se înlocuiască toate aparîtiile unui sir de caractere notat **șir1** prin alt sir de caractere notat **șir2**. Cele două siruri de caractere vor fi citite separat, înaintea textului.
19. Scrieți un program care să citească de la tastatură două propoziții terminate prin caracterul punct și apoi să afișeze o listă a literelor ce apar în ambele propoziții.
20. Fie declarația:

```
struct lista
{
    int nr;
    float nota;
};
```

Să se scrie o funcție care să aibă ca parametru un tablou cu maximum 100 de elemente de tip **lista** și să returneze ca rezultat valoarea câmpului **nr** asociată celei mai mari valori a câmpului **nota** din tablou.

21. Se consideră un tablou de elemente ce conțin informații despre activitatea de producție a mai multor firme. Fiecare element memorează următoarele

informații: codul numeric al firmei, codul numeric al produsului, cantitatea produsă. Să se scrie o funcție care, folosind ca parametri tabloul și o valoare numerică  $x$ , furnizează ca rezultat numărul de firme care produc cea mai mare cantitate de produse al căror cod numeric este  $x$ .

**22.** Definiți un tip structură pentru descrierea unui moment de timp exprimat prin valorile oră, minute, secunde. Scrieți apoi o funcție care să calculeze timpul scurs între două momente de timp cunoscute. De exemplu, de la momentul 3:45:15 până la momentul 9:44:03 au trecut 5 ore, 58 de minute și 48 de secunde. Atenție la momentele de timp situate înainte și după miezul nopții !

**23.** Definiți un tip structură (eventual structură ierarhizată) pentru descrierea unui moment prin dată calendaristică (zi, lună, an) și moment de timp (oră, minute, secunde). Scrieți apoi o funcție care la fiecare apelare să actualizeze momentul de timp curent prin incrementare cu o secundă și o funcție care să realizeze o operație similară, dar prin incrementare cu o zi a datei calendaristice. Cele două funcții vor fi utilizate de o a treia astfel: se va apela funcția de actualizare a momentului de timp și, dacă în acest mod se depășește miezul nopții, se va apela a doua funcție pentru a se trece la ziua următoare. Rezultatul final va reprezenta structura completă, actualizată.

**24.** Definiți un tip structură convenabil pentru descrierea unei figuri geometrice plane. De exemplu, structura va conține denumirea formei figurii geometrice și:

- pentru un cerc: raza;
- pentru un dreptunghi : dimensiunile celor două laturi.

Scrieți apoi o funcție care să aibă ca rezultat aria unei figuri geometrice date și folosiți-o într-un program.

**25.** Să se definească un tip structură care să permită declararea de variabile numere complexe și cu ajutorul acesteia să se simuleze în limbajul C toate operațiile asupra numerelor complexe: adunarea, scăderea, înmulțirea,

împărțirea, calcularea modulului, a argumentului, a părții reale și a părții imaginare. De asemenea, să se scrie funcții care să citească de la tastatură o valoare complexă și, respectiv, să afișeze pe ecran o astfel de valoare.

26. Scrieți o funcție care să furnizeze ca rezultat puterea a patra a unei valori de tip real. Folosiți funcția pentru a calcula valoarea expresiei  $(a+b)^4$ , unde  $a$  și  $b$  sunt două valori de tip real.
27. Scrieți o funcție care să aibă ca rezultat valoarea minimă existentă într-un tablou de numere.
28. Să se scrie o funcție care să determine cel mai mic multiplu comun pentru două numere întregi precizate.
29. Scrieți o funcție care să aibă ca rezultat suma cifrelor ce formează un număr întreg.
30. Să se scrie o funcție **cifra(n,m)** care are ca rezultat valoarea celei de-a  $m$ -a cifre de la dreapta spre stânga a numărului  $n$  scris în sistemul zecimal. De exemplu: **cifra(7283, 3)** are valoarea 2.
31. Scrieți o funcție care să stabilească dacă un număr dat  $n$  conține în reprezentarea sa zecimală o anumită cifră precizată, notată, de exemplu, c. Se va utiliza apoi această funcție pentru a afișa toți întregii cu valori cuprinse între 1 și p (p citit de la tastatură) pentru care numărul, pătratul și cubul reprezentării sale conțin aceeași cifră. Exemple de astfel de numere: 1, 5, 6, 10, 11, 12 etc.
32. Scrieți o funcție pentru calcularea valorii  $x^n$ , unde  $x$  și  $n$  sunt numere întregi pozitive folosite ca parametri. Utilizați apoi această funcție în cadrul unui program pentru a testa dacă valoarea  $4(k+1)$  divide sau nu suma  $(2k+1)^{2k+3} + (2k+3)^{2k+1}$ , considerând  $k$  o valoare cunoscută ce îndeplinește condiția  $0 \leq k \leq 5$ .

33. Să se scrie o funcție care să calculeze suma elementelor unui tablou de numere.
34. Definiți o funcție care să stabilească dacă o valoare dată se află printre cele  $n$  elemente ale unui sir de numere dat. Folosiți apoi această funcție pentru a crea un sir de elemente numerice distincte pe baza unor valori citite de la tastatură.
35. Să se definească o funcție care inserează într-un sir numeric dat ce conține deja  $n$  elemente ordonate crescător un nou element, astfel încât sirul obținut să fie în continuare ordonat crescător. Se va utiliza apoi această funcție pentru a comoda două siruri  $a$  și  $b$ , având  $p$ , respectiv  $q$  elemente ordonate crescător, într-un singur sir  $a$ , ordonat de asemenea crescător.
36. Într-o matrice dată, notată  $a$ , cu  $l$  linii și  $c$  coloane să se permute circular dreapta fiecare linie  $i$  cu  $i$  poziții. Se va utiliza o funcție care permută circular dreapta cu o poziție componentele unui vector.
37. Dându-se doi vectori  $x$  și  $y$  având  $p$ , respectiv  $q$  componente reale, să se creeze vectorii intersecție, reuniune și diferență a celor doi vectori. Se va defini și utiliza o funcție care stabilește dacă o valoare aparține sau nu unui vector  $b$  având  $n$  componente.
38. Pentru un număr natural  $n$  dat, se cere:
- să se determine toți divizorii pozitivi
  - să se calculeze numărul divizorilor
  - să se calculeze suma divizorilor
  - să se calculeze produsul divizorilor numărului  $n$
39. Să se scrie un program care ordonează crescător elementele de pe diagonala principală unei matrice  $A$  cu  $n$  linii și  $n$  coloane prin interschimbarea liniilor și coloanelor matricei. Se vor folosi două funcții: o funcție care

interschimbă două linii ale căror indici sunt dați ca parametri și o funcție care interschimbă două coloane ale căror indici sunt dați ca parametri.

*Exemplu:* Matricea  $\begin{pmatrix} 4 & 5 & 2 \\ 1 & 9 & 7 \\ 8 & 6 & 3 \end{pmatrix}$  devine  $\begin{pmatrix} 3 & 8 & 6 \\ 2 & 4 & 5 \\ 7 & 1 & 9 \end{pmatrix}$

**40.** Să se tipărească toate numerele prime din intervalul  $[a,b]$ . Se va folosi o funcție care verifică dacă un număr natural dat ca parametru este sau nu număr prim.

*Exemplu:* Pentru  $a=5$  și  $b=40$  se vor afișa valorile  $5, 7, 11, 13, 17, 19, 23, 29, 31, 37$ .

**41.** Să se tipărească numerele din intervalul  $[a,b]$  ce se divid cu suma cifrelor lor. Se va folosi o funcție care returnează suma cifrelor unui număr natural dat ca parametru.

*Exemplu:* Pentru  $a=20$ ,  $b=50$  se afișeaza valorile  $20, 21, 24, 27, 30, 36, 40, 42, 45, 48, 50$ .

**42.** Dintre primele  $n$  numere naturale, să se afișeze acelea cu proprietatea că suma cifrelor lor este impară. Se va folosi o funcție care returnează suma cifrelor unui număr natural dat ca parametru.

*Exemplu:* Dacă  $n=30$  se obțin numerele  $1, 3, 5, 7, 9, 10, 12, 14, 16, 18, 21, 23, 25, 27, 29$ .

**43.** Scrieți o funcție care verifică dacă un număr este palindrom sau nu. Funcția va avea:

- a) un parametru de tip int
- b) un parametru de tip sir de caractere

*Exemplu:* 16561 este palindrom.

**44.** Să se scrie o funcție care determină a m-a cifra a numărului n. Numerotarea se face de la dreapta începând cu 0.

*Exemplu:* Dacă n=31724 și m=3 se afișează cifra 1.

**45.** Să se scrie un program care modifică un vector astfel încât el să conțină doar valori distincte. Se va folosi o funcție care verifică dacă o valoare există deja în vector.

*Exemplu:*  $x=\{10,3,2,10,7,9,3,4,10\} \Rightarrow x=\{10,3,2,7,9,4\}$ .

**46.** Se dă matricea A cu m linii și n coloane. Să se permute circular spre dreapta cu k poziții toate liniile matricei. Se va folosi o funcție care permută circular spre dreapta cu o poziție o linie a matricei.

*Exemplu:* Dacă matricea este  $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$  și  $k=2$  rezultă  $\begin{pmatrix} 3 & 4 & 1 & 2 \\ 7 & 8 & 5 & 6 \end{pmatrix}$ .

**47.** Se citește de la tastatură o matrice A cu m linii și n coloane și elemente numere întregi. Să se determine linia (liniile) din matrice care conține cele mai multe elemente nenule. Se va folosi o funcție care returnează numărul elementelor nenule de pe o linie a cărui indice i se transmite ca parametru.

*Exemplu:* Pentru matricea  $\begin{pmatrix} 1 & 3 & 2 & 0 \\ 5 & 0 & 0 & 9 \\ 7 & 0 & 10 & 1 \end{pmatrix}$  liniile cerute sunt 1 și 3.

**48.** Se dă un vector x cu n componente numere întregi. Să se calculeze:

- suma componentelor din vectorul x
- numărul componentelor impare din x
- elementul maxim din vector

*Exemplu:* Pentru  $x = \{2,9,15, 6,3,12 ,66,8,5\} \Rightarrow$  a) 126; b) 4; c) 66.

**49.** Elementele unei matrice A cu m linii și 2 coloane sunt coordonatele a m puncte în plan  $p_1, p_2, \dots, p_m$ ,  $m > 3$ . Să se afișeze numerele de ordine ale celor trei linii care determine triunghiul de arie maximă, precum și valoarea acestei arii. Se va folosi o funcție care returnează aria unui triunghi de laturi date.

**50.** Se dă un vector cu numere întregi. Să se afișeze toate perechile de numere prime între ele. Se va folosi o funcție care determină cel mai mare divizor comun a două numere.

**51.** Se dau  $m, n \in \mathbb{N}$  și mulțimile  $A = \{a_1, a_2, \dots, a_m\}$  și  $B = \{b_1, b_2, \dots, b_n\}$  memorate ca vectori. Să se scrie câte un program care calculează:

- a)  $C = A \cap B$
- b)  $C = A \cup B$
- c)  $C = A - B$

Vectorii A și B se consideră nesortați. Se va folosi o funcție care verifică dacă o anumită valoare se găsește într-un vector sau nu.

*Exemplu:*

Dacă  $A=\{1,5,2,9,12\}$  și  $B=\{5,3,4,6,33,99,12,64\}$  atunci:

- a)  $C=\{5,12\};$
- b)  $C=\{1,5,2,9,12,3,4,6,33,99,64\};$
- c)  $C=\{1,2,9\}.$

**52.** Se consideră doi vectori A cu m elemente și B cu n elemente. Se cere să se determine reuniunea celor doi vectori folosind:

- o funcție pentru ordonarea unui vector;
- o funcție pentru a insera o valoare într-un vector ordonat pentru a rămâne ordonat (se va folosi căutarea binară a poziției de inserare).

*Exemplu:* Dacă  $A = \{1, 5, 2, 9, 12\}$  și  $B = \{5, 3, 4, 6, 33, 99, 12, 64\}$  atunci  $A \cup B = \{1, 5, 2, 9, 12, 3, 4, 6, 33, 99, 64\}$ .

**53.** Se citește de la tastatură un sir de caractere. Să se verifice dacă acesta conține doar caractere mici sau mari ale alfabetului englez și în caz afirmativ afișează mulțimea literelor distincte din sirul dat.

*Exemplu:* Pentru sirul ‘acesta este un sir’ răspunsul este afirmativ iar mulțimea solicitată este ‘a’, ‘c’, ‘e’, ‘l’, ‘n’, ‘r’, ‘s’, ‘t’, ‘u’.

**54.** Să se scrie câte o funcție pentru a calcula:

- a) ultima cifră a numărului  $2^x$ , pentru  $x$  număr natural
- b) ultima cifră a numărului  $a^b$ , cu  $a$  și  $b$  cifre ale sistemului zecimal
- c) ultima cifră a sumei  $2^a + 3^a + \dots + 9^a$ , cu  $a$  număr natural
- d) ultima cifră a numărului  $a^b$ , pentru  $a$  și  $b$  numere naturale mai mici sau egale cu 32000, citite de la tastatură
- e) ultima cifră a sumei  $1^n + 2^n + \dots + n^n$ , pentru  $n$  număr întreg citit de la tastatură

*Exemplu:* a)  $x=15 \Rightarrow 8$ ;

- b)  $a=3, b=9 \Rightarrow 3$ ;
- c)  $a=11 \Rightarrow 4$ ;
- d)  $a=123, b=547 \Rightarrow 7$ ;
- e)  $n=11 \Rightarrow 6$ .

**55.** Se dă o matrice patratică  $A$ , de dimensiune  $n$ . Să se scrie un program care afișează matricele:  $A, A^2, A^3, \dots, A^n$ .

**56.** Pentru codificarea unui text format din litere mici se folosesc două tablouri bidimensionale  $5 \times 5$ , generate prin program.

$a$	$b$	$c$	$d$	$e$		1	2	3	4	5
$f$	$g$	$h$	$i$	$j$		6	7	8	9	10
$k$	$l$	$m$	$n$	$p$		11	12	13	14	15
$o$	$r$	$s$	$t$	$u$		16	17	18	19	20
$v$	$w$	$x$	$y$	$z$		21	22	23	24	25

Codificarea se realizează caracter după caracter astfel:

- caracterul din linia i și coloana j se codifică prin elementul corespunzător din tabloul de numere întregi
- după efectuarea acestei codificări, linia i din tabloul de numere se permute circular spre dreapta cu i poziții, apoi coloana j din același tabel se permute circular în sus cu j poziții

Să se scrie un program care, pentru orice text dat conținând doar caractere mici ale alfabetului englez, să afișeze succesiunea de numere obținută prin codificare. Textul poate să conțină și spații care se vor ignora. Între două codificări succesive se va lăsa exact un spațiu.

*Exemplu:* Codificarea sirului ‘exemplu’ este 5 23 4 2 12 11 20.

**57.** Simulați jocul de Bingo. Jucătorul introduce o variantă jucată, iar calculatorul generează aleator varianta câștigătoare și afișează rezultatul (linie, bingo, nimic). Varianta jucată și cea câștigătoare se vor afișa pe ecran colorate, punându-se în evidență cu culori intermitente linia sau cartonul câștigător, dacă varianta jucată este câștigătoare.

**58.** Se consideră o *matrice rară* (adică cu foarte multe elemente nule) A(n,n) cu p elemente nenule. Matricea este memorată economic sub forma a doi vectori v(i), t(i),  $1 \leq i \leq p$ , care rețin valorile elementelor nenule ale matricei, respectiv poziția acestor elemente, liniarizate pe linii. Să se scrie un program care realizează următoarele operații, fiecare fiind implementată cu ajutorul unei funcții separate:

a) citind vectorii v și T refac matricea A;

*Exemplu:* Pentru  $n=5$ ,  $p=8$ ,  $v = \{2,7,8,4,6,1,3,5\}$ ,  $T = \{1,5, 9,11,14,18,20,22\}$

se va afișa matricea  $A = \begin{pmatrix} 2 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 8 & 0 \\ 4 & 0 & 0 & 6 & 0 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 5 & 0 & 0 & 0 \end{pmatrix}$

- b) determină vectorii  $v$  și  $T$  corespunzători matricei transpusă a matricei  $A$ , fără a reconstitui matricea

*Exemplu:* Transpusa matricei anterioare este dată prin  $n=5$ ,  $V = \{ 2, 4, 5, 1, 8, 6, 7, 3 \}$  și  $T = \{ 1, 3, 10, 14, 17, 18, 21, 4 \}$ .

- c) considerând matricele  $A(n,n)$  și  $B(n,n)$  cu  $p$  respectiv  $q$  elemente nenule, calculează suma celor două matrice folosind memorarea economică;

*Exemplu:* Pentru  $n=5$  și matricele date prin  $V1 = \{ 2, 7, 8, 4, 6, 1, 3, 5 \}$ ,  $T1 = \{ 1, 5, 9, 11, 14, 18, 20, 22 \}$  și respectiv  $V2 = \{ 1, 5, 1, 3, 4, 4, 2, 9 \}$ ,  $T2 = \{ 1, 2, 8, 10, 12, 16, 18, 24 \}$  se obține  $V = \{ 3, 5, 7, 1, 8, 3, 4, 4, 6, 4, 3, 3, 5, 9 \}$  și  $T = \{ 1, 2, 5, 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24 \}$ .

- d) același enunț ca la punctul anterior pentru produsul celor două matrice.

*Exemplu:* Pentru  $n=5$  și matricele date prin  $V1 = \{ 2, 7, 8, 4, 6, 1, 3, 5 \}$ ,  $T1 = \{ 1, 5, 9, 11, 14, 18, 20, 22 \}$  și respectiv  $V2 = \{ 1, 5, 1, 3, 4, 4, 2, 9 \}$ ,  $T2 = \{ 1, 2, 8, 10, 12, 16, 18, 24 \}$  se obține  $V = \{ 2, 10, 63, 32, 16, 28, 20, 12, 4, 27, 5, 15 \}$  și  $T = \{ 1, 2, 4, 6, 8, 11, 12, 13, 17, 19, 23, 25 \}$ .

Observație. La toate punctele se consideră că elementele nenule se vor citi/afișa în ordinea liniarizării pe linii a matricei (adică în ordinea  $a_{11}, a_{12}, \dots, a_{1n}, a_{21}, \dots, a_{2n}, \dots, a_{nn}$ , deci vectorul  $T$  este sortat crescător).

## 4.4. Teste grilă



4.4.1. Se consideră un vector v care poate memora maxim 25 de numere întregi, din care folosim efectiv numai primele n elemente v[0] ,v[1],...,v[n-1] (unde n este variabil). Scrieți antetul unei funcții numită calcul, care primește drept parametri vectorul v împreună cu numărul său de elemente n și nu returnează nimic.

- a) void calcul (int v[n], 25)
- b) void calcul (int v[25] , int n)
- c) void calcul (int v[ ] , int n)
- d) void calcul (int v[ ] , n)
- e) void calcul (int v, int n)



4.4.2. Considerăm o funcție demo, de tipul void (nu returnează nimic), pentru care se definesc ca parametri trei variabile întregi. Cum realizăm apelul funcției, astfel încât la apel să dăm ca parametri variabilele întregi a, b și c ?

- a) demo(int a, int b, int c);
- b) demo(int a; int b; int c);
- c) demo (a, b, c);
- d) demo(a;b;c);
- e) demo (int a, b, c);



4.4.3. Scrieți o funcție D care primește ca parametru un număr întreg a și returnează valoarea lui a+2.

a)  
int D (int a);  
{ D=(a+2); }

b)  
int D (int a) {  
D=a+2; }

c)  
int D (int a) ;  
{ return(a+2) ; }

d)  
int D (int a)  
{ return a+2; }

- e) Nici una dintre variantele anterioare



#### 4.4.4. Care dintre afirmațiile de mai jos sunt adevărate ?

- a) Parametrii definiți în antetul unei funcții se numesc actuali, iar cei care apar la apelul funcției se numesc formali
- b) Valoarea returnată de către o funcție poate fi transmisă ca parametru altor funcții
- c) Variabilele de tip tablou nu se pot transmite ca parametri funcților
- d) Variabilele globale sunt cunoscute pe tot parcursul programului în care au fost declarate în toate modulele care urmează declarației
- e) Corpul unei funcții trebuie cuprins între "{" și "}", numai dacă este alcătuit din cel puțin două instrucțiuni distințe



#### 4.4.5. Avantajele utilizării funcțiilor într-un program sunt:

- a) Se poate obține o economisire a spațiului de memorie rezervat variabilelor folosite în cadrul programului
- b) O viteză mai mare în execuția programului
- c) Posibilitatea de a executa de mai multe ori instrucțiunile cuprinse într-o funcție
- d) Un program care conține funcții poate fi urmărit și corectat mai ușor
- e) Nici unul dintre avantajele de mai sus



#### 4.4.6. Deduceți sirul de valori care se afișează în urma execuției programului următor.

- a) 100,10,10000,10,10000,10
- b) 100, 0, 100, 0, 100, 0
- c) 0, 0, 0, 0, 0, -10
- d) 100, 0, 10000, 0, 16960, 0
- e) 100, 0, 0, 0, 10000, -10

```
#include <stdio.h>
int s, f;
int sf (int a)
{
    f = a ;
    return (a*a) ;
}
int main()
{
    f=10; s = sf (f) ;
    printf ("%6d%6d", s, f) ;
    f=10; s=sf(10); s*=sf(f);
    printf ("%6d%6d", s, f) ;
    f=10; s=sf(f); s*=sf(10);
    printf ("%6d%6d", s, f) ;
```



4.4.7. Determinați valorile pe care le afișează programul de mai jos:

- a) 43, 22, 10
- b) 43, 12, 10
- c) 47, 10, 12
- d) 44, 22, 11
- e) 44, 12, 11

```
#include<stdio.h>
int x, y;
int T(int m, int n)
{
    m=n+x;
    n+=1;
    return (n+y+m);
}
int main()
{
    y=10;
    x=12 ;
    printf("%3d ",T(x,y));
    printf("%3d%3d ",x,y);
}
```



4.4.8. Fie programul:

```
#include<stdio.h>
void F (.....)
{
    a+=2;
    b--;
    c=a+b;
}
int main ()
{
    int x,y,z;
    x=2;
    y=4;
    F(x,y,z);
    printf("%3d%3d%3d",x,y,z);
}
```

Funcția F primește ca parametri trei numere întregi a, b și c. Cum trebuie scris antetul complet al funcției, astfel încât programul să afișeze, în ordine, valorile 447?

- a) void F (int a, int b, int c)
- b) void F (int a, int &b, int &c)
- c) void F (int&a, int b, int &c)
- d) void F (int &a, int b, into)
- e) void F (int &a, int &b, int &c)



#### 4.4.9. Fie programele următoare:

```
//Programul P1
#include <stdio.h>
void apel(int &m, int n)
{
    m=m+n;
    n=(m-n)/2;
}
int main()
{
    int x,y; x=11;
    y=3;
    do{
        apel(x,y);
        printf("x=%d y=%d", x, y);
    }while( cond );
}
```

```
//Programul P2
#include <stdio.h>
void apel (int &ia, int &n)
{
    m=m+n ;
    n=(m-n)/2;
}
int main()
{
    int x,y;
    x=11; y=3;
    while ( cond )
    {
        apel(x,y);
        printf("%3d%3d", x, y) ;
    }
}
```

Care dintre următoarele afirmații este adevărată ?

- a) În situația în care condiția (expresia logică) cond este "y>0", programul P1 va intra într-un ciclu infinit
- b) În situația în care condiția (expresia logică) cond este "y<10", programul P2 va afișa opt valori
- c) Dacă înlocuim cond cu "y<=0" în ambele programe, ele vor deveni echivalente (adică vor produce aceleași rezultate)
- d) Dacă înlocuim cond cu "y<=0" în programul P1, acesta va afișa valorile x=14 și y=5
- e) Dacă înlocuim cond cu "y<5" în programul P2, acesta va afișa valorile x=14 și y=5



4.4.10. Ce valoare trebuie citită în variabila m astfel, încât programul următor să afișeze valoarea 4?

- a) 12
- b) 13
- c) 14
- d) 15
- e) 16

```
#include <stdio.h>
void F(int &nr, int x)
{
    do{
        x=x/2;
        nr++;
    } while (x>0);
int main()
{
    int m,n;
    scanf ("%d", &m);
    n=0;
    F(n,m);
    printf ("%d", n);
}
```



4.4.11. Precizați, care dintre funcțiile de mai jos returnează: 1 dacă numărul x este prim, respectiv, 0 în caz contrar.

a)  
int p(int x)  
{  
 int i, ok; for(i=2;i<x;i++)  
 if (x%i==0) ok=0;  
 else ok=1;  
 return ok;  
}

b) int p (int x)  
{  
 int i ;  
 for(i=2; i<x; i++)  
 if (x%i==0) return 0;  
 else return 1;  
}

c)  
int p (int x)  
{  
 int i , ok=1 ;  
 for(i=2;i<x;i++)  
 if (x%i==0) ok=0;  
 return ok;  
}

d)  
int p (int x)  
{  
 int i;  
 return 1;  
 for (i=2; i<x; i++)  
 if (x%i==0) return 0 ;  
}

e) Toate funcțiile anterioare



4.4.12. Se consideră următoarea structură de program:

```
#include <iostream.h>
void main()
{
    int v[20],i,n;
    .....
    calcul (v,2.5,10);
}
void calcul (int v[], float x, int a)
{
    .....
}
```

Funcția calcul, fiind scrisă după main, are nevoie de un prototip care trebuie plasat înaintea lui main. Cum poate arăta acest prototip?

- a) void calcul (int v, float x, int a);
- b) void calcul (int v[ ], float x, int a);
- c) void calcul (int, float, int);
- d) void calcul (int [ ], float, int);
- e) void calcul (int \*, float, int);



4.4.13. Deducreți ce valori va afișa, în ordine, programul următor?

- a) -3 2 7 -1 0 1
- b) 0 0 0 -1 0 1
- c) -3 2 7 0 0 0
- d) -1 0 1 -3 2 7
- e) 0 0 0 -3 2 7

```

#include <iostream.h>
int u[5], v[5], w[5], t[5], i;
void calcul (int *w, int *t)
{
    for(i=0;i<3;i++)
    {
        w[i] = 0;
        t[i] = v[i] - u[i];
    }
}
int main ()
{
    for (i=0 ; i<3 ; i++)
    {
        u[i] = 2*i-1;
        v[i] = 3*i-2;
        w[i] = u[i] + v[i];
        t[i] = 0;
    }
    calcul (w, t) ;
    for(i=0;i<3;i++) cout<<" "<<w[i] ;
    for(i=0;i<3;i++) cout<<" "<<t[i] ;
}

```



- 4.4.14. Ce valoare se va afișa în urma execuției programului următor?
- a) 7
  - b) 8
  - c) 9
  - d) 10
  - e) 11

```

#include <iostream.h>
int f(int a=5, int b=2)
{
    return --a + 2*b++;
}
int main()
{
    cout<<f();
}

```



4.4.15. Se consideră următorul program:

```
int f(int x, int y=5)
{
    static int m=3;
    m+=x;
    return m>=y ? m : -1;
}
int main()
{
    int a,b,c;
    a = f(4);
    b = f(1,7);
    c = f();
}
```

Care dintre cele trei apeluri realizate în main sunt corecte?

- a) Numai primele două apeluri sunt corecte iar valorile variabilelor a și b vor fi a=7, b=8
- b) Numai primele două apeluri sunt corecte iar valorile variabilelor a și b vor fi a=7, b=-1
- c) Numai al doilea apel este corect, și în urma execuției lui se va obține b=8
- d) Numai al doilea apel este corect, și în urma execuției lui se va obține b=-1
- e) Toate cele trei apeluri sunt corecte



4.4.16. Fie funcția F de mai jos:

```
int * F (int *a, int *b, int c)
{
    d=(*a+*b+c)/3;
    return &d ;
}
```

Presupunem că sunt declarate următoarele variabile:

```
int *x, *y, z, *m, d;
```

Care dintre secvențele de program de mai jos nu afișează corect valoarea returnată de către funcția F ?

a)  
scanf("%d %d %d",x,y,&z);  
printf("\n%d",\*F(x,y,z));

b)  
cin>>\*x>>\*y>>z;  
cout<<\*F(x,y,z);

c)  
\*m = \*F(x,y,z);  
cout<<\*m;

d)  
n=\*F(x,y,z);  
cout<<n;

e)  
cout<<\*F(2,3,4);



4.4.17. Fie funcția:

```
void afis1(int v[])
{
    int i=0 ;
    while (i<4)
        cout<<v[i++]<<"  ";
```

Funcțiile afis2 și afis3 de mai jos sunt corecte și echivalente cu funcția dată afis1 ? (adică afișează același sir de valori ca și funcția afis1)

```
void afis2(int v[4])
{
    int i=0;
    do
        cout<<*(v+i++)<<"  ";
    while (i<4);
}
```

```
void afis3(int v[4])
{
    int i,*x[4];
    for (i=0;i<4;i++)
    {
        x[i]=&v[i];
        cout<<x[i]<<"  ";
    }
}
```

- a) Ambele funcții afis2 și afis3 conțin erori
- b) Ambele funcții afis2 și afis3 sunt corecte și echivalente cu afis1

- c) Ambele funcții afis2 și afis3 sunt corecte, dar nici una dintre ele nu este echivalentă cu afis1
- d) Funcția afis2 este corectă și echivalentă cu afis1, iar afis3 conține erori
- e) Funcția afis3 este corectă și echivalentă cu afis1, iar afis2 conține erori



4.4.18. Care va fi valoarea lui n în urma execuției programului următor:

- a) n va avea valoarea -3
- b) n va avea valoarea -4
- c) n nu va avea în nici un caz valorile de la a) sau b)
- d) n va avea valoarea -3.56
- e) Programul conține erori de sintaxă

```
#include <iostream.h>
int *intreg (float *p)
{
    return (int *)p;
}
int main()
{
    int n;
    float x=-3.56;
    n=x;
    n=*intreg(&x);
}
```



4.4.19. Ce valori se vor afișa în urma execuției programului următor?

- a) programul este eronat
- b) 0 și 9
- c) 5 și 0
- d) 5 și 9
- e) 0 și 0

```
#include <iostream.h>
int q;
int *A(int *m, int *n)
{
    *n=*(m+1);
    *m=*(n+1);
    q=*(m-1);
    return &q;
}
void B(int *p)
{
    int y=4;
    y=*(A(&y,p));
}
int main()
{
    int y=9,x;
    B(&x);
    cout<<x<<" "<<y;
}
```



4.4.20. Fie funcțiile f1 și f2 având prototipurile:

```
void f1 (int (*p) [8]);  
void f2 (int p [4] [8]);
```

Analizați corectitudinea apelurilor celor două funcții în următorul program:

```
int main ()  
{  
    int m[4] [8] , (*q) [8],*r[8];  
    f1(m); f2(m);  
    f1(q); f2 (q);  
    f1(r); f2 (r);  
}
```

- a) Numai primele patru apeluri sunt corecte
- b) Numai primele două apeluri sunt corecte
- c) Numai primele două apeluri sunt greșite
- d) Toate apelurile sunt greșite
- e) Toate apelurile sunt corecte



4.4.21. Ce reprezintă declarația următoare:

```
int *(*f) (int *);
```

- a) Este greșită sintactic
- b) Funcție care primește argument pointer la întreg și întoarce pointer la întreg
- c) Pointer către întreg
- d) Pointer către funcție care are argument pointer către întreg și întoarce pointer către întreg
- e) O altă semnificație decât cele de la b), c) și d)



#### 4.4.22. Fie programul:

```
#include<iostream.h>
int plus (int x)
{
    return x++;
}
int minus (int x)
{
    return --x;
}
void test (int p, int q,int (*F1) (int) ,int (*F2) (int))
{
    if (p && q) cout<<F1(p);
    else cout<<F2(q) ;
}
int main()
{
    int a=2, b=5;
    test (a, b, plus, minus);
    test (3, 0, plus, minus);
    test (a, b, plus (a) , minus (b) );
    test (3, 0, plus (3) , minus (0) );
}
```

Ce se poate spune despre apelurile funcției **test** realizate din **main** ?

- a) Primele două apeluri sunt corecte, determinând afişarea valorilor 2, respectiv, -1, iar ultimele două sunt eronate
- b) Primele două apeluri sunt corecte, determinând afişarea valorilor 3, respectiv, -1, iar ultimele două sunt eronate
- c) Ultimele două apeluri sunt corecte, determinând afişarea valorilor 2, respectiv, -1, iar primele două sunt eronate
- d) Ultimele două apeluri sunt corecte, determinând afişarea valorilor 3, respectiv, -1, iar primele două sunt eronate
- e) Toate cele patru apeluri sunt eronate



4.4.23. Ce valoare se va afișa în urma execuției programului următor?

- a) 21
- b) 39
- c) 18
- d) 30
- e) antetul funcției este greșit

```
#include <iostream.h>
int f(int n,...)
{
    int *a=&n+1, *b=a+2;
    return n*(^a+*b);
}
int main()
{
    cout<<f(3, 2, 11, 5, 8);
}
```



4.4.24. Ce trebuie pus în locul secvenței XXX astfel încât programul de mai jos să furnizeze minimul unui număr variabil de argumente, precizat la apel prin valoarea lui n?

```
#include <stdio.h>
double f(int n,...)
{
    double min=*(double *)(&n+1);
    for(int i=1;i<n;i++)
        if (XXX<min) min=XXX;
    return min;
}
int main()
{
    int n=3;
    float a=2,b=8,c=-1;
    printf("%f",f(n,a,b,c));
    printf("%f",f(n,2.0,8.0,-1.0));
    printf("%f", f (3,2.0,8.0,-1.0));
}
```

- a) \*(double \*) (&n+1+i)
- b) \*((double \*)(&n+1)+i)
- c) \* (double \*)(&n+(1+i))
- d) \* (double \*) (& (n+1)+i)
- e) O altă construcție decât cele anterioare



4.4.25. Precizați ce va afișa programul de mai jos:

- a) Programul nu va afișa nimic pentru că este greșit
- b) "aac"
- c) "abc"
- d) "aba"
- e) "aabc"

```
#include <stdio.h>
char &f(char *p,int n)
{
    return *(p+n);
}
int main()
{
    char *s="abc"; f(s,1)='a';
    printf("%s",s);
}
```



4.4.26. Care dintre apelurile de funcție de mai jos va schimba valoarea lui n?

- a) Ambele
- b) Nici una
- c) I
- d) II
- e) Antetul funcției e greșit

```
#include <stdio.h>
int *f(int &n)
{
    return &n;
}
int main()
{
    int n=7;
    f(n)=2;           // (I)
    *f(n)=3;          // (II)
}
```



4.4.27. Analizați programul de mai jos și precizați care dintre afirmațiile date sunt *false*.

```

int &f1(int &x)
{
    return x=1;
}
int *f2 (int *p)
{
    int a=2;
    f1(a);
    *p=3;
    return p;
}
int main()
{
    int b=4;
    f1(b);
    f1(b)=5;
    f2(&b);
    *f2(&b)=6;
}

```

- a) În urma execuției liniei `f1(b)` , valoarea variabilei b va fi 4
- b) În urma execuției liniei `f1(b)=5`, valoarea variabilei b va fi 5
- c) În urma execuției liniei `f2 (&b)`, valoarea variabilei b va fi 3
- d) În urma execuției liniei `*f2 (&b)`, valoarea variabilei b va fi 6
- e) Valoarea variabilei locale a din funcția f2 nu poate fi modificată prin nici un mecanism în funcția principală main()



#### 4.4.28. Se dau următoarele programe:

```

// (I)
char &f (char &a)
{
    if (a>96) return a=a-32;
    else return a=a+32;
}
int main()
{
    char x='x';
    f(x)=x;
}

```

```

// (II)
char &f(char &a)
{
    if (a>96) return a-32;
    else return a+32;
}
int main()
{
    char x=' x';
    f(x)=x;
}

```

Alegeți răspunsurile corecte:

- a) Ambele programe conțin erori
- b) Primul program este corect și în final x va avea valoarea 'x'
- c) Al doilea program este corect și în final x va avea valoarea 'x'
- d) Ambele programe fac același lucru
- e) Ambele programe sunt corecte, dar nu fac același lucru



4.4.29. Precizați valoarea variabilei n rezultată în urma execuției programului:

```
int f(char a[2])
{
    int i=0;
    while (a[i++]>); return i;
}
int main()
{
    int n=f("abcdefg");
}
```

- a) 0
- b) 2
- c) 9
- d) Corpul funcției conține o buclă infinită
- e) Programul are erori de sintaxă

## 4.5. Răspunsuri la întrebările grilă

- 4.5.1. b) și c)
- 4.5.2. c)
- 4.5.3. c) și d)
- 4.5.4. b) și d)
- 4.5.5. a), c) și d)
- 4.5.6. e)
- 4.5.7. b)
- 4.5.8. c)
- 4.5.9. a), b) și e)
- 4.5.10. a), b), c) și d)
- 4.5.11. c)
- 4.5.12. d)
- 4.5.13. b)
- 4.5.14. b)
- 4.5.15. a)
- 4.5.16. e)
- 4.5.17. b)
- 4.5.18. c)
- 4.5.19. b)
- 4.5.20. a)
- 4.5.21. d)
- 4.5.22. a)
- 4.5.23. a)
- 4.5.24. b)
- 4.5.25. b)
- 4.5.26. d)
- 4.5.27. a)
- 4.5.28. b)
- 4.5.29. c)

# Pointeri

## 5.1. Declarare

**U**n **pointer** este o variabilă care are ca valoare adresa unui obiect cu care operează limbajul C (variabilă, constantă, funcție).

Folosirea pointerilor este determinată de două motive și anume: este singura modalitate de a efectua anumite calcule și în plus folosirea pointerilor duce la generarea unui cod sursă de program mai eficient și mai compact. Astfel putem enumera situațiile când limbajul C folosește pointerii, și anume: atunci când se transmite unei funcții o matrice sau un sir de caractere, sau când vrem să transmitem parametri prin referință, adică atunci când vrem să modificăm variabilele respective.

Declararea unei variabile de tip pointer se face astfel:



**tip \*nume\_pointer;**

Unde:

- **tip** reprezintă oricare dintre tipurile limbajului C și indică tipul variabilei a cărei adresă este memorată de variabila pointer.
- **nume\_pointer** reprezintă numele ales de utilizator pentru a identifica pointerul respectiv

- operatorul '\*' este obligatoriu la declarare



Exemple:

```
char *pc; // pointer de tip caracter  
int *pi; // pointer de tip întreg  
float *pf; // pointer de tip real
```

Observație: Pot exista și pointeri fără tip, care se declară astfel:



```
void *nume_pointer;
```

Aceștia se utilizează pentru ca în program să primească valoarea oricărui tip de dată.

## 5.2. Operatori specifici pointerilor

Operatorii folosiți pentru lucrul cu pointerii sunt operatorii unari:

1. Operatorul ' & ' (operator de adresă) – se aplică unei variabile furnizând adresa acelei variabile.

Exemplu: Următorul program arată cum se poate folosi operatorul de adresă pentru a afișa adresele unor variabile de tipuri diferite.



```
#include<stdio.h>  
int main(void)  
{  
    int indice=1;  
    float salariu_dolari=20000.0;  
    long salariu_lei=30000000L;  
    printf("Adresa variabilei indice este %x\n",&indice);  
    printf("Adresa variabilei salariu_dolari este %x\n",  
    &salariu_dolari);  
    printf("Adresa variabilei salariu_lei este  
    %x\n",&salariu_lei);  
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

*Adresa variabilei indice este fff4*

*Adresa variabilei salariu\_dolari este fff0*

*Adresa variabilei salariu\_lei este ffec*

Observație: Nu poate fi aplicat expresiilor, constantelor sau variabilelor de tip regisztr.

**Inițializarea pointerilor** se poate face cu o valoare de adresă de memorie, prin folosirea operatorului de adresă ‘&’:



**nume\_pointer = &variabila;**

Exemplu: Următorul program declară o variabilă de tip pointer și atribuie adresa unei variabile de tip întreg și apoi afișează valoarea varaiabilei pointer împreună cu adresa variabilei de tip întreg.



```
#include<stdio.h>
int main(void)
{
    int indice = 1;
    int *p_indice;
    p_indice = &indice;
    printf("Valoarea lui p_indice %x\nValoarea lui indice
%d\n Adresa lui indice este %x\n", p_indice, indice, &indice);
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

*Valoarea lui p\_indice fff4*

*Valoarea lui indice 1*

*Adresa lui indice este fff4*

2. **Operatorul ‘\*’ (operator de redirectare)** – se aplică pointerilor și furnizează obiectul referit de acel pointer.

Exemplu: Următorul program atribuie pointerului adresa unei variabile de tip întreg, afișează adresa sa împreună cu valoarea memorată la adresa pe care o are pointerul, apoi modifică valoarea variabilei prin intermediul pointerului și o afișează.



```
#include<stdio.h>
int main(void)
{
    int indice = 100;
    int *p_indice;
    p_indice = &indice;
    printf("Adresa lui p_indice %x\nValoarea la p_indice
%d\n", p_indice, *p_indice);
    *p_indice = 50;
    printf("Valoarea variabilei indice este %d\n", indice);
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

*Adresa lui p\_indice fff4*

*Valoarea la p\_indice 100*

*Valoarea variabilei indice este 50*

### 5.3. Aritmetica pointerilor

Așa cum am spus mai înainte, un pointer reține o adresă de memorie la care se află o valoare de un anumit tip, mai exact un pointer este o variabilă care reține o locație de memorie. Cum locațiile de memorie sunt zone contigue, dacă se adaugă valoarea 1 unui pointer, atunci el va reține următoarea locație de memorie, dacă se adaugă 10, atunci el va reține locația se memorie aflată la distanța de 10 locații de memorie de cea inițială. Totuși aritmetica pointerilor depinde în principal de tipul de variabilă a cărei adresă de memorie o reține. Astfel, dacă avem un pointer care conține adresa 2000 a unei variabile de tip *char*, atunci prin adăugarea unei valori de 1 vom obține adresa 2001, deoarece o variabilă de tip *char* se reține pe un octet de memorie. Dacă pointerul ar fi conținut aceeași adresă de memorie, dar a unei

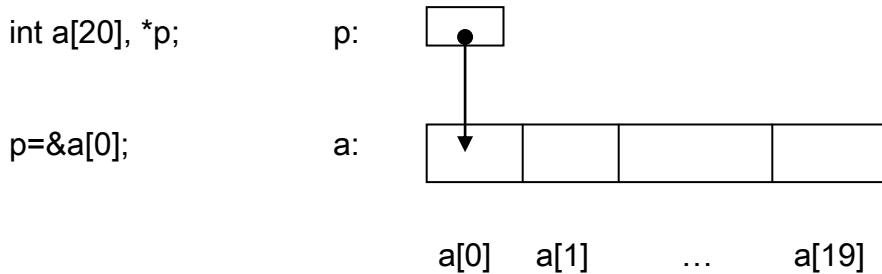
variabile de tip *int*, atunci prin adăugarea unei valori de 1 am obținere adresa 2002, la fel în cazul unei variabile de tip *float*, am obținere adresa 2004.

### 5.3.1. Pointeri și tablouri

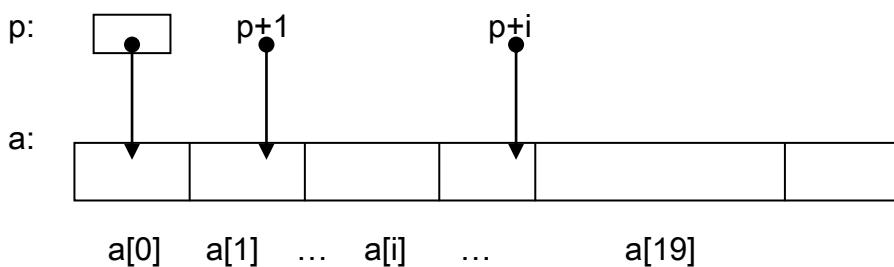
În limbajul C++ există o stânsă legătură între pointeri și tablouri, astfel încât orice problemă care se poate rezolva cu tablouri, poate fi rezolvată și cu pointeri.

Putem spune că un pointer care reține adresa unui tablou, reține de fapt adresa primului element al tabloului.

Exemplu:



*Dacă p este un pointer către un element al unui tablou oarecare, atunci  $p+1$  este un pointer către elementul următor al tabloului,  $p+i$  se referă la elementul al i-lea, iar  $p-i$  la elementul al i-lea înainte de p.*



Observație: În expresia `p+i` ne apare faptul că exemplificăm pe un tablou cu elemente de tip întreg. De unde putem concluziona că:

*Dacă p este un pointer către un obiect (tip\_object \*p), atunci  $p+i$  va fi un pointer către al i-lea obiect aflat după obiectul referit de p.*

Numele unei variabile de tip tablou este de fapt adresa primului element (adică adresa elementului 0 al tabloului).

Deci,  $p = \&a[0]$ ;  $\Leftrightarrow p = a$ ;

Dacă  $p$  și  $q$  sunt doi pointeri către elemente ale aceluiași tablou (adică  $p = \&a[i]$  și  $q = \&a[j]$ ), atunci putem efectua următoarele operații:



### 1) Comparații:

$p == q$  // adică se compară  $i$  cu  $j$   
 $p != q$   
 $p < q$   
 $p <= q$   
 $p > q$   
 $p >= q$

2)  $p - q$  // adică numărul de elemente ale tabloului, care se află între elementul referit de  $p$  și elementul referit de  $q$

Exemplu: Următorul program afișează cu ajutorul unor tablouri de tipuri diferite, diferența între doi pointeri.



```
#include <stdio.h>
int main(void)
{
    char *s="Informatica", *cp, *cq;
    int tab_i[]={1, 2, 3, 4}, *ip, *iq;
    long tab_l[]={5, 6, 7}, *lp, *lq;
    float tab_f[]={1.2, 3.44, 5.43, 6.890, 2.33}, *fp, *fq;
    double tab_d[]={1.11, 2.222, 3.333, 4.44}, *dp, *dq;
    cp=cq=s;
    cp++; printf("cp-cq=%d", cp-cq);
    ip=iq=tab_i;
    ip++; printf("ip-iq=%d", ip-iq);
    lp=lq=tab_l;
    lp++; printf("lp-lq=%d", lp-lq);
    fp=fq=tab_f;
    fp++; printf("fp-fq=%d", fp-fq);
    dp=dq=tab_d;
    dp++; printf("dp-dq=%d", dp-dq);
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

$$cp - cq = 1$$

$$ip - iq = 1$$

$$lp - lq = 1$$

$$fp - fq = 1$$

$$dp - dq = 1$$

### 5.3.2. Echivalențe de scriere

Avem următoarele echivalențe de scriere:



$$\begin{aligned} p = \&a[0] &\Leftrightarrow & p = a \\ p + 1 &\Leftrightarrow \&a[1] \\ *(p+1) &\Leftrightarrow a[1] \\ p + i &\Leftrightarrow \&a[i] \\ a + 1 &\Leftrightarrow \&a[1] \\ *(a+1) &\Leftrightarrow a[1] \\ a + i &\Leftrightarrow \&a[i] \\ *(a + i) &\Leftrightarrow a[i] \end{aligned}$$

Deci, putem scrie:

$$a[i] \Leftrightarrow *(a+i) \Leftrightarrow *(i+a) \Leftrightarrow i[a]$$

Din echivalențele de mai sus, putem desprinde o regulă a compilatorului C++, și anume:

Orice expresie de forma  $a[i]$  este transformată imediat într-o expresie  $*(a+i)$ , adică o expresie în care apare un **pointer** (adică  $a$ ) și un **deplasament** (adică  $i$ ).

Exemplu: Următorul program prezintă ultima regulă obținută mai sus și anume  $a[i] \Leftrightarrow i[a]$ .



```
#include<stdio.h>
int main(void)
{
    int tablou_i[10] = {0,1,2,3,4,5,6,7,8,9};
    int i = 7;
    printf("%d\n", tablou_i[i]);
    printf("%d\n", i[tablou_i]);
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

7  
7

Observație: Succesiunea operatorilor ‘ &\* ’ aplicată unui pointer cu deplasament are ca efect, la primul pas, selectarea conținutului (valorii) de la respectiva adresă, iar la al doilea pas, selectarea adresei respectivului conținut. Deci aplicând această succesiune de operatori se obține tot obiectul. Nu același lucru se poate spune despre succesiunea ‘ \*& ’.

Exemplu: Următorul program arată echivalența dintre un tablou și un pointer care îl referă.



```
#include<stdio.h>
int main(void)
{
    int tablou_i[10] = {0,1,2,3,4,5,6,7,8,9};
    int *p = tablou_i;
    printf("%x %x\n", tablou_i+3, &*(tablou_i+3));
    printf("%x %x\n", p+3, &*(p+3));
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

ffe8 ffe8  
ffe8 ffe8

### 5.3.3. Folosirea pointerilor de tip *void\**

Limbajul C++ permite folosirea pointerilor de tip *void\**, pe post de pointeri de tip universal, astfel încât, atunci când se utilizează aceștia, trebuie convertiți la tipul variabilei prin operatorul de conversie explicită *cast*.

Adresa 0 de memorie este definiță prin valoarea **NULL** pentru pointeri.

Exemplu: Următorul program prezintă interschimbarea a două variabile de tip întreg prin intermediul a doi pointeri de tip *void\**.



```
#include<stdio.h>
int main(void)
{
    int a=10,b=5;
    void *pa,*pb,*temp;
    (int*)pa=&a; (int*)pb=&b;

    *(int*)temp=*(int*)pa;
    *(int*)pa=*(int*)pb;
    *(int*)pb=*(int*)temp;

    printf("\na = %d si b = %d\n",a,b);
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

*a = 10 si b = 5*

### 5.3.4. Expresii compacte cu pointeri

Cu ajutorul pointerilor se pot scrie expresii compacte în care pointerii apar împreună cu operatorii de *incrementare* ( *++* ) și *decrementare* ( *--* ). Astfel pot apărea următoarele situații:



EXPRESIA	OPERATIA	CE SE MODIFICĂ
*p++	postincrementare	pointerul
*p--	postdecrementare	pointerul
*++p	preincrementare	pointerul
*--p	predecrementare	pointerul
++*p	preincrementare	obiectul
--*p	predecrementare	obiectul
(*p)++	postincrementare	obiectul
(*p)--	postdecrementare	obiectul

*Expresiile din prima grupă modifică pointerul și nu obiectul la care se referă acesta.* Acest tip de expresii îmbunătățesc viteza de execuție a unui program și se recomandă utilizarea acestora în ciclurile repetitive.

*Expresiile din a doua grupă modifică obiectul referit, astfel încât operatorii ++ și -- sunt aplicați de către compilator asupra obiectului și nu asupra pointerului.* Deci, nu putem utiliza în aceste expresii un pointer la o structură, uniune sau o funcție.

Exemplu: Următorul program prezintă modul în care acționează operatorii ++ și -- asupra expresiilor compacte cu pointeri.



```
#include <stdio.h>
int main(void)
{
    int a = 1, b = 2, c = 0;
    int *x = &a, *y = &b;
    printf("%d\t", ++(*x));
    printf("%d\t", (*y)--);
    printf("%d\n", ++c);
    a++; b=2; c-=(*x) + b++;
    printf("%d\t", ++(*x));
    printf("%d\t", (*y)--);
    printf("%d\n", ++c);
    x=y;
    printf("%d\t", ++(*x));
    printf("%d\t", (*y)--);
    printf("%d\n", ++c);
    printf("%d\t%d\t%d\n", a, b, c);
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

2	2	1
4	3	-3
3	3	-2
4	2	-2

La prima afișare, `++(*x)` este preincrementare asupra obiectului referit de `x` și deci variabila a se modifică ( se afișează valoarea 2 ), `(*y)--` este postdecrementare și deci variabila `b` nu se modifică înainte de afișare ( se afișează valoarea 2 ), iar `++c` este preincrementare și deci se modifică variabila `c` ( se afișează valoarea 1 ). Continuând raționamentul și la celelalte afișări se obțin valorile respective.

### 5.3.5. Tablouri de pointeri

Pointerii pot fi grupați în tablouri declarându-i astfel:



**tip \*nume\_tablou\_pointeri[];**



Exemplu:

```
char *tpc[10]; // tablou de 10 pointeri către caractere  
float *tpf[25]; // tablou de 25 pointeri către numere reale
```

Acest tip de tablou conține adrese către valori de un anumit tip.

Exemplu: Următorul program afișează un tablou de pointeri către siruri de caractere și le ordonează alfabetic.



```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char *tp[10] = {"1234567890", "abcde", "ABCDE", "info",
"INFO", "limbaj", "Programare", "C/C++", "Pascal",
"nota10"};
    int i,inv;
    char *t;
    printf("Afisarea sirului de pointeri dat\n");
    for(i=0;i<=9;i++)
        if(*tp[i]!=NULL) printf("%s\n",tp[i]);
    do
    {
        inv=0;
        for(i=0;i<9;i++)
            if(strcmp(tp[i],tp[i+1])>0)
            {
                t=tp[i];tp[i]=tp[i+1];tp[i+1]=t;
                inv=1;
            }
    }while(inv==1);
    printf("Sirul de pointeri ordonat alfabetic\n");
    for(i=0;i<=9;i++)
        if(*tp[i]!=NULL) printf("%s\n",tp[i]);
}
```

După compilarea și execuția programului, pe ecran vor apărea următoarele valori:

*Afisarea sirului de pointeri dat*

*1234567890*

*abcde*

*ABCDE*

*info*

*INFO*

*limbaj*

*Programare*

*C/C++*

*Pascal*

*nota10*

*Sirul de pointeri ordonat alfabetic*

*1234567890*

*ABCDE*

*C/C++*

*INFO*

*Pascal*

*Programare*

*abcde*

*info*

*limbaj*

*nota10*

## 5.4. Probleme rezolvate

### 5.4.1. Enunțuri

1. Transformarea unui număr dintr-o bază oarecare  $b$  în baza 10.

Funcția ***strtol*** este folosită pentru a face conversia unui *număr* dintr-o bază  $b$  ( $2 \leq b \leq 36$ ) în baza 10. Numărul inițial, întrucât poate conține și litere, este reprezentat ca un sir de caractere. Pentru verificarea corectitudinii, este setat un pointer la primul caracter din sir care a general eroarea de conversie. Dacă pointerul este NULL, nu a apărut nici o eroare.

2. Să se determine poziția relativă a două drepte. Pentru poziția relativă a două drepte se vor trata cazurile :

1) Dreptele sunt paralele

2) Dreptele sunt concurente. În particular se va trata cazul cand dreptele sunt perpendiculare.

3) Dreptele coincid

Pentru aceasta vom calcula coeficienții fiecăreia din cele două drepte. În acest scop a fost definită funcția *traiectorie* care plecând de la coordonatele a două puncte distincte, calculează parametrii dreptei care trece prin cele două puncte. După obținerea celor 6 coeficienți  $a_1, b_1, c_1$  și  $a_2, b_2, c_2$ , vom avea cazurile:

-  $a_1 \cdot b_2 = a_2 \cdot b_1$ : dreptele sunt paralele sau confundate. Dacă  $a_1 \cdot c_2 = a_2 \cdot c_1$  atunci dreptele sunt confundate, altfel dreptele sunt paralele.

- În cazul când dreptele sunt perpendicular, atunci conform definiției trebuie să avem:  $a_1 \cdot a_2 = b_1 \cdot b_2$

3. Să se facă un program de determinare a unei date, știind o dată de pornire și un număr de zile trecute.

Pentru a se ține cont de numărul diferit de zile ale lunilor anului se va defini un tablou de constante și se va lua în calcul separat cazul când anul este bisect, luna februarie având 29 de zile.

4. Se dau  $n$  numere reale ce reprezintă rădăcinile unui polinom. Să se determine coeficienții acestui polinom.

Soluție:

Fie  $a_1, a_2, \dots, a_n$  rădăcinile reale ale polinomului  $P$ :

$$P = (x-a_1) * (x-a_2) * \dots * (x-a_n)$$

Fie polinomul  $P$  de forma:

$$P = b_n * x^n + b_{n-1} * x^{n-1} + \dots + b_1 * x + b_0$$

Coefficienții  $b_0, b_1, \dots, b_n$  se pot calcula fie cu ajutorul **relațiilor lui Viete**, fie calculând produsul tuturor termenilor ce formează descompunerea.

Astfel, dacă:

$$Q = c_m * x^m + c_{m-1} * x^{m-1} + \dots + c_1 * x + c_0 \text{ și } R = Q * (x - a_{m+1})$$

unde  $R = d_{m+1} * x^{m+1} + d_m * x^m + \dots + d_1 * x + d_0$  vom avea relațiile:

$$d_{m+1} = c_m$$

$$d_k = c_{k-1} - c_k * a_{m+1}, 1 \leq k \leq m$$

$$d_0 = -c_0 * a_{m+1}$$

5. Fiind dat un sir de caractere reprezentând un text, să se scrie un program care să facă înlocuirea de un număr de ori a aparițiilor unui cuvânt cu altul.

Putem face înlocuirea unui sir fără să folosim facilitățile funcțiilor din biblioteca string.h. Înlocuirea unui subșir într-un sir se va face definind funcția **inlocuirি** care are parametrii:

*sir*, textul inițial

*d*, cuvântul inițial

*c2*, cuvântul care se va înlocui

*nr*, numărul maxim de înlocuire

## 5.4.2. Soluții propuse

### Problema 1:

Transformarea unui număr dintr-o bază oarecare  $b$  în baza 10.

Functia *strtol* este folosita pentru a face conversia unui *număr* dintr-o bază  $b$  ( $2 \leq b \leq 36$ ) în baza 10. Numărul inițial, întrucât poate conține și litere, este reprezentat ca un șir de caractere. Pentru verificarea corectitudinii este setat un pointer la primul caracter din șir care a generat eroarea de conversie. Dacă pointerul este NULL, nu a apărut nici o eroare.



```
#include<stdio.h>
#include<stdlib.h>
{
    int baza;
    char numar[30], *p;
    long nr;
    puts("\n Introduceti baza: ");
    scanf("%d", &baza);
    puts("\n Introduceti numarul in baza %d : ", baza);
    scanf("%s", numar);
    nr = strtol(numar, &p, baza);
    if(p)
        printf("Eroare ka caracterul %c", *p);
    else
        printf("Numarul este : %ld", nr);
}
```

### Problema 2:

Să se determine poziția relativă a două drepte. Pentru poziția relativă a două drepte se vor trata cazurile:

- 1) Dreptele sunt paralele
- 2) Dreptele sunt concurente. În particular se va trata cazul când dreptele sunt perpendiculare
- 3) Dreptele coincid

Pentru aceasta vom calcula coeficienții fiecăreia din cele două drepte. În acest scop, a fost definită funcția *traiectorie* care plecând de la coordonatele a două puncte distincte, calculează parametrii dreptei care trece prin cele două

puncte. După obținerea celor 6 coeficienți  $a_1$ ,  $b_1$ ,  $c_1$  și  $a_2$ ,  $b_2$ ,  $c_2$ , vom avea cazurile:

- $a_1 \cdot b_2 = a_2 \cdot b_1$ : dreptele sunt paralele sau confundate. Dacă  $a_1 \cdot c_2 = a_2 \cdot c_1$ , atunci dreptele sunt confundate, altfel dreptele sunt paralele.
- În cazul când dreptele sunt perpendiculare atunci conform definiției trebuie să avem:  $a_1 \cdot a_2 = b_1 \cdot b_2$



```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<graphics.h>
void traiercie(int x1, int x2, int y1, int y2, double *a, double *b,
double *c) ;
int x1, int x2, int x3, int y1, int y2, int y3 ;
double a1,b1,a2,b2,c1,c2,x,y ;
void traiercie(int x1, int x2, int y1, int y2, double *a, double *b,
double *c)
{
    *a=y2-y1;
    *b=x1-x2;
    *c=x2*y1-x1*y2 ;
}
int main(void)
{
    puts("Introduceti coordonatele capetelor celor 2 drepte");
    scanf("%d %d %d %d",&x1,&y1,&x2,&y2) ;
    scanf ("%d %d %d %d",&x3,&y3,&x4,&y4);
    traiercie(x1,x2,y1,y2, &a1,&b1,&c1);
    traiercie(x3, x4,y3,y4, &a2, &b2, &c2);
    if (a1*b2==a2*b1)
        if (a1*c2==a2*c1) printf("Dreptele se confunda" );
        else printf("Drepte paralele ");
    else
    {
        x=(c2*b1-c1*b2) / (a1*b2-a2*b1) ;
        y=(a2*c1-a1*c2) / (a1*b2-a2*b1) ;
        printf(" Dreptele sunt concurente in (%lg,%lg)",x,y);
        if(a1*a2==b1*b2) printf(" si perpendiculare");
        else printf(" si neperpendiculare.");
    }
}
```

### Problema 3:

Să se facă un program de determinare a unei date știind o dată de pornire și un număr de zile trecute. Pentru a se ține cont de numărul diferit de zile ale lunilor anului, se va defini un tablou de constante și se va lua în calcul separat cazul când anul este bisect, luna februarie având 29 de zile.



```
#include<stdio .h>
#include<conio .h>
const int t[13]={ 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
void calcul (int z, int an, int nr_zile, int *zi, int *luna, int *a){
    int i;
    for(i=1;i<=12;i++)
        if ( (i==2) && (an%4==0) )
            if( nr_zile>29 )      nr_zile-=29;
            else break;
        else
            if( nr_zile>t[i] ) nr_zile-=t[i];
            else break;
    *a=an;
    do{
        if( (*an%4==0) && (i==2) ) {
            z-=29-nr_zile;
            nr_zile=0;
            i++;
            if(z<=0) { i--; z+=29; break; }
        }
        else{
            if(i>12) { (*a)++; i++; }
            z-=t[i]-nr_zile; nr_zile=0; i++;
            if(z<=0) { i--; z+=t[i]; break; }
        }
    }while(z>0); *luna=i; *zi=z;
}
int main(void)
{
    int nr_zile, zi, an, z, luna, a, i;
    printf("\nIntroduceti data (an luna zi) : ");
    scanf("%d%d%d", &an, &luna, &z);
    for (a=0,i=2;i<luna;i++) a+=t[ i]+(an%4==0 && i==2);
    if(luna<1 || luna>12 || z>t[luna] +(an%4==0 && i==2))
        { printf("Data gresita");return;}
    z+=a;
    printf("Introduceti numarul de zile trecute: ");
    scanf("%d",&nr_zile);
    calcul(z, an, nr_zile, &zi, &luna, &a);
    printf("Anul: %d \t Luna: %d \t Ziua: %d", a, luna, zi);
}
```

#### Problema 4:

Se dau  $n$  numere reale ce reprezintă rădăcinile unui polinom. Să se determine coeficienții acestui polinom.

Soluție:

Fie  $a_1, a_2, \dots, a_n$  rădăcinile reale ale polinomului  $P$ :

$$P = (x-a_1) * (x-a_2) * \dots * (x-a_n)$$

Fie polinomul  $P$  de forma:

$$P = b_n * x^n + b_{n-1} * x^{n-1} + \dots + b_1 * x + b_0$$

Coefficienții  $b_0, b_1, \dots, b_n$  se pot calcula fie cu ajutorul **relațiilor lui Viète**, fie calculând produsul tuturor termenilor ce formează descompunerea.

Astfel, dacă:

$$Q = c_m * x^m + c_{m-1} * x^{m-1} + \dots + c_1 * x + c_0 \text{ și } R = Q * (x - a_{m+1})$$

unde  $R = d_{m+1} * x^{m+1} + d_m * x^m + \dots + d_1 * x + d_0$  vom avea relațiile:

$$d_{m+1} = c_m$$

$$d_k = c_{k-1} - c_k * a_{m+1}, \quad 1 \leq k \leq m$$

$$d_0 = -c_0 * a_{m+1}$$



```
#include <stdio.h>
#include <stdlib.h>

int cit_date(float **rad)
{
    int n, i;
    printf("n= ");
    scanf("%d", &n);
    *rad = malloc((n+1) * sizeof(float));
    for (i=1; i <=n ; i++)
    {
        printf("Radacina [%d] = ", i);
        scanf("%f", &(*rad)[i]);
    }
    return n;
}
```

```

float* polinom(float *a, int n)
{
    float *b;
    int m, k;
    b = malloc((n+1) * sizeof(float));
    b[1]=1;
    b[0]=- a[n];
    for( m = 2; m <= n; m++)
    {
        b[m] = b[m-1];
        for(k=m-1; k>0; k--)
            b[k]=b[k-1] - b[k] * a[m];
        b[0] = -b[0] * a[m];
    }
    return b;
}
void afis_polinom(float *coef, int grad)
{
    int i;
    printf("P= ");
    for(i=grad; i>0 ; i++)
        if(coef[i] != 0)
            printf("%.4f *X^%d+", coef[i], i);
    if(coef[0] != 0 || grad == 0)
        printf("%.4f", coef[0]);
    printf("\n");
}
int main(void)
{
    float *a;
    float *b;
    int n = cit_date(&a);
    b = polinom(a,n);
    afis_polinom(b,n);
}

```

### Problema 5:

Fiind dat un sir de caractere reprezentand un text, sa se scrie un program care sa faca inlocuirea de un numar de ori a aparitiilor unui cuvant cu altul.

Putem face inlocuirea unui sir fara sa folosim facilitatile functiilor din biblioteca string.h. Inlocuirea unui subsir intr-un sir se va face definind functia *inlocuire* care are parametrii:

*sir*, textul initial

*d*, cuvântul inițial

*c2*, cuvântul care se va înlocui

*nr*, numărul maxim de înlocuiră



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char *inlocuiriri(char *,char *,char '*',int );
int main(void)
{
    int nr;
    char sir[255], c1[200], c2[200] ;
    puts("Introduceti sirul:"); gets(sir);
    puts("Introduceti cuvantul care va fi inlocuit:"); gets(c1);
    puts("Introduceti cuvantul inlocuitor:"); gets(c2);
    puts("Numarul maxim de inlocuiră:"); scanf("%d",&nr);
    puts("\nNoul sir:"); puts(inlocuiriri(sir,c1,c2,nr));
}
char *inlocuiriri(char *sir,char *c1,char *c2,int nr)
{
    char *m,* p, q[100], a[100];
    int n, poz, i=0;
    m=sir;
    do{
        p=strstr(m,c1);
        n=strlen(sir) ;
        if (p != NULL)
        {
            nr--;
            poz=n-strlen(p);
            for(i=0; i<poz; i++)
                *(q+i)=*(sir+i);
            for(i=poz; i<=n-strlen(c1); i++)
                q[i]=sir[i+strlen(c1)];
            for(i=0; i<poz; i++)
                *(a+i)= *(q+i);
            for(i=poz; i<=poz+strlen(c2);i++)
                a[i]=c2[i-poz];
            for(i=poz+strlen(c2); i<=strlen(c2)+n+1; i++)
                a[i]=q[i-strlen(c2)];
            a[i]='\0';
            m=p+1;
            sir=strdup(a);
        }
    }while(p && nr);
    return a;
}
```

## 5.5. Probleme propuse spre rezolvare

1. Se dau multimi prin intermediul a doi vectori  $x$  și  $y$ , cu  $n$  ( $1 \leq n \leq 100$ ) componente întregi. Se cere să se verifice dacă aceste multimi sunt proporționale. În caz afirmativ se va afișa și sirul de rapoarte egale.

2. Se dau  $m, n \in N$  și multimile  $A = \{a_1, a_2, \dots, a_m\}$  și  $B = \{b_1, b_2, \dots, b_n\}$  memorate ca vectori. Să se scrie câte un program care calculează :

- a)  $C = A \cap B$
- b)  $C = A \cup B$
- c)  $C = A - B$

Cei doi vectori se consideră sortați crescător.

3. Fie  $A$  o matrice patratică de ordinul  $n$ . Să se scrie un program care generează o matrice o matrice  $B$  (patratică de ordinul  $n$ ) ale cărei elemente sunt definite prin relația  $b_{ij} = (a_{ij} + a_{ji}) / 2$  efectuând un număr minim de pași.

4. Se citește un text  $T$  de la tastatură și două cuvinte  $c1$  și  $c2$ . Să se scrie un program care tipărește textul obținut din  $T$  prin înlocuirea aparițiilor cuvântului  $c1$  cu cuvântul  $c2$ . Cuvintele se consideră separate prin unul sau mai multe caractere ",", ".", ";" și spațiu.

5. Se citește de la tastatură un cuvânt de lungime cel mult 20 de caractere, format numai din litere mari. Să se afișeze toate cuvintele distințe ce se pot forma prin eliminarea câte unui singur caracter din cuvântul dat.

6. Se citesc  $n$  numere naturale de cel mult 9 cifre fiecare. Să se afișeze cifrele comune tuturor numerelor.

7. Se citesc de la tastatură pe rând n cuvinte formate din litere mici ale alfabetului englez. Să se afișeze perechile de cuvinte din cele citite cu proprietatea că cele două cuvinte folosesc aceeași mulțime de litere distințe.
8. Se dă un vector de n numere întregi. Să se afișeze numerele din vector care au cele mai multe cifre egale c (citit de la tastatură) în scrierea lor în baza b ( $b \leq 9$ ). Se va folosi o funcție care returnează numărul de cifre c din scrierea în baza b a unui număr.
9. Să se detrezine toate numerele întregi de trei cifre  $\overline{abc}$  cu proprietatea că  $\overline{abc} = a^3 + b^3 + c^3$ .
10. Se dă un număr real pozitiv x și un număr întreg n ( $n > 6$ ). Să se calculeze  $x^n$  folosind mai puțin de  $n-1$  înmulțiri.
11. Scrieți un program de ordonare descrescătoare a unui tablou de numere unidimensional (vector) folosind ca instrument de lucru pointeri.
12. Scrieți o funcție numită **sort3** care să ordeneze crescător trei numere întregi date, fără a folosi tipul tablou.
13. Numim *linii gemene* într-o matrice, două linii care conțin exact aceleași elemente, eventual într-o altă dispozitie pe coloane. Scrieți un program care determină toate perechile de linii gemene dintr-o matrice A dată, cu m linii și n coloane, și elemente numere întregi. Pentru fiecare astfel de pereche, se vor afișa pe un rând indicii liniilor componente.

Exemplu :  $A = \begin{pmatrix} 1 & 5 & 2 & 4 \\ 3 & 9 & 7 & 2 \\ 2 & 5 & 4 & 1 \end{pmatrix}$  singura pereche este (1,3).

14. Se dă o matrice A cu m linii și n coloane. Să se eliminate din A liniile și coloanele la intersecția cărora se află un număr natural al cărui scriere în baza 2 are cel puțin două cifre egale cu 1. Se va folosi o funcție care returnează numărul de cifre egale cu 1 dintr-un număr, o funcție de eliminare a unei liniilor dintr-o matrice și o funcție care elimină o coloană dintr-o matrice.

15. Se introduce de la tastatură o cifră k diferita de 0 și 1 și un text x cu cel mult 255 caractere, format din litere mici ale alfabetului englez. Textul x se codifică literă cu literă, fiecare literă fiind înlocuită cu un număr natural egal cu  $k^{poz}$ , unde poz este poziția literei în alfabet. Astfel 'a' se codifică prin  $k^1$ , 'b' se codifică prin  $k^2$ , și.a.m.d. Între numerele codificării se lasă un spațiu liber. Se cere să se afișeze codificarea textului. Se va folosi o funcție care returnează puterea  $a^b$ , și o funcție care returnează poziția unei litere în cadrul alfabetului.

16. Se consideră funcția f definită prin:

$$f = \begin{cases} x^2 + y^2 & \text{daca } \max(|x|, |y|) < 1 \\ \frac{1}{x} + \frac{1}{y} & \text{daca } \min(|x|, |y|) > 1 \\ x + y & \text{altfel} \end{cases}$$

Se citesc n perechi (x,y). Să se afișeze cea mai mare respectiv cea mai mică valoare a funcției f.

17. Scrieți un program care, pentru doi întregi x și n, afișează valoarea expresiei:

$$S = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + \frac{x^{2n-1}}{(2n-1)!}$$

Se vor folosi trei funcții: o funcție pentru ridicarea la putere, o funcție de calcul al factorialului și o funcție care primind ca parametrii valorile x și n, returnează valoarea expresiei S.

18. Să se scrie un program care determină suma componentelor prime dintr-un vector de numere întregi. Se va folosi o funcție care verifică dacă un număr este prim sau nu.

*Exemplu:* Pentru vectorul  $x = \{5, 9, 2, 8, 18, 3\}$  suma ceruta este  $s=10$ .

19. Se dă un vector de  $n$  numere întregi. Să se afișeze numerele din vector care au cele mai multe cifre egale cu  $c$  (citit de la tastatură) în scrierea lor în baza  $b$  ( $b \leq 9$ ). Se va folosi o funcție care returnează numărul de cifre  $c$  din scrierea în baza  $b$  a unui număr.
20. Se citește de la tastatură un text format din litere mici ale alfabetului englez precum și separatorii punct (.), virgula (,), punct și virgula (;), spațiu ( ) și două puncte (:). Se cere să se scrie un program care formează toate grupurile de cuvinte în care fiecare cuvânt din grup reprezintă o *anagramă* a celorlalte cuvinte din grup. Nu se vor afișa grupurile dintr-un singur cuvânt. Se va folosi o funcție care returnează primul cuvânt dintr-un text și o funcție care verifică dacă un cuvânt este anagrama altui cuvânt.
21. Se citesc  $n$  numere naturale de la tastatură. Să se afișeze numerele din vector pe grupe, o grupă având numere cu aceeași *suma de control*. Grupele se vor afișa în ordinea descrescătoare a numărului de elemente din grupă. Dacă două grupe au același număr de componente ele se vor afișa în ordinea crescătoare a valorii cifrei de control corespunzătoare grupei respective. Suma de control a unui număr se obține calculând suma cifrelor numărului, apoi suma cifrelor acestei cifre și.a.m.d. până se obține o sumă formată dintr-o singură cifră care va reprezenta cifra de control. (Suma de control a numărului 183 este  $1+8+3 = 12 = 3$ ).

*Exemplu:* Pentru  $x = \{156, 91, 12, 10081, 822, 75\}$  se va afisa;

Grupa 1: 156,12,822,75

Grupa 2: 10081

Grupa 3: 91

22. Să se scrie un program care tipărește un număr precizat de termeni din sirul 1, 2, 1, 3, 2, 1, 4, 2, 2, 5, 4, 3, 2, 1, 6, 2, 2, 3, 3, 3... obținut din sirul

numerelor naturale prin înlocuirea fiecărui număr natural printr-un grup de numere. Numărul prim  $p$  este înlocuit prin numerele  $p, p-1, \dots, 3, 2, 1$  iar numărul compus  $n$  este înlocuit prin  $n$  urmat de toți divizorii săi proprii, un divizor  $d$  repetându-se de  $d$  ori.

**23.** Realizați, folosind tablouri, o aplicație practică referitoare la un concurs de admitere într-un liceu cu un singur profil. Există  $n$  elevi candidați, iar numărul de locuri este  $m$ . Fiecare elev ia două note. Primii  $m$  elevi sunt considerați admisi dacă fiecare notă a lor este mai mare sau cel puțin egală cu 5. Aplicația va permite, aşadar, ordonarea elevilor după media obținută, dar se cere și o ordonare alfabetică, precum și implementarea căutării unui elev după nume. Pentru fiecare din aceste operații, precum și pentru citirea datelor și afișarea diferitelor situații se va implementa câte o funcție. Alegerea opțiunilor se va face de către utilizator prin intermediul unui mic "meniu".

**24.** Un număr natural este perfect dacă este egal cu suma divizorilor săi diferenți de el însuși. Să se determine toate numerele perfecte mai mici decât  $n$ ,  $n$  fiind citit de la tastatură. Se vor folosi următoarele funcții:

- funcție de determinare a divizorilor unui număr întreg
- funcție care verifică dacă un număr este perfect sau nu

*Exemplu:* Dacă  $n=500$  se afișează valorile 6, 28, 496.

**25.** Dându-se un vector neordonat  $A$  cu  $n$  elemente, să se găsească cel de-al  $k$ -lea element din tabloul ordonat, fără a-l ordona. Se va folosi o funcție care determină poziția minimului elementelor unui sir.

Indicație. Determinarea minimului sirului dat realizează de fapt găsirea primului termen al sirului ordonat fără a-l ordona. Dacă pe poziția acestui minim punem o valoare mai mare decât a tuturor termenilor, la următorul pas determinarea minimului va realiza stabilirea celui de-al doilea termen al sirului ordonat și.a.m.d.

**26.** Printre numerele mai mici sau egale cu un număr  $n$  dat, să se găsească cel care are cei mai mulți divizori. Se va folosi o funcție ce calculează numărul de divizori ai unui număr.

*Exemplu:* Pentru  $n=100$ , numărul cerut este 60 care are 12 divizori.

**27.** Să se calculeze cel mai mare divizor comun a două numere date  $m$  și  $n$  astfel:

- pentru fiecare număr se creează un tablou cu toți divizorii acestuia inclusiv divizorii banali (1 și numărul însuși)
- se selecteză apoi într-un tablou toți divizorii comuni și se ia cel mai mare dintre aceștia

Se va folosi programarea modulară.

**28.** *Numerele puternice* sunt acele numere naturale care au numărul divizorilor mai mare decât numărul divizorilor oricărui număr natural mai mic decât numărul respectiv. Programul cere să se afișeze al  $n$ -lea număr puternic ( $n < 70$ ).

*Exemplu:* Cel mai mic număr puternic este 1, urmează 2, 4 și 6.

**29.** Să se scrie o funcție cu numele *max2* care determină maximul dintre valorile a două variabile de tip întreg, apoi să se scrie o funcție cu numele *max3* care determină maximul dintre valorile a trei variabile de tip întreg astfel încât să utilizeze funcția *max2*.

**30.** Să se scrie o funcție *perm2* care realizează interschimbarea valorilor a două variabile și să se scrie apoi o funcție *ord* care ordonează crescător un sir de numere întregi astfel încât să apeleze *perm2*.

**31.** Scrieți un program care să afișeze calendarul pe un an, dându-se anul și ziua din săptămâna în care cade 1 ianuarie. Veți folosi o funcție care să afișeze calendarul pentru o lună, dându-se ziua din săptămâna cu care o luna începe și numărul de zile din lună. Fiecare lună succesivă începe cu ziua din

luna ce urmează după ultima zi a lunii precedente. Zilele săptămânii vor fi numerotate de la 0 la 6 pentru Duminica până la Sâmbăta. A nu se uita de anii bisecți.

32. Se dă un tablou bidimensional cu m linii și n coloane având componente numere naturale. Să se determine cel mai mare divizor comun și cel mai mic multiplu comun al componentelor tabloului.

33. Pentru un număr natural n dat, să se stabilească raportul dintre numărul divizorilor pari și numărul divizorilor impari ai numărului n.

*Exemplu:* Pentru n=246 raportul este 1 (4 divizori pari și 4 divizori impari).

34. Se citesc de la tastatură k matrice cu m linii și n coloane. Să se determine câte din aceste matrice verifică cel puțin una din următoarele proprietăți:

- a) există cel puțin o linie cu toate elementele pozitive
- b) există cel puțin o linie cu toate elementele în progresie geometrică
- c) există cel puțin o linie cu toate elementele ordonate crescător

35. Să se rezolve în mulțimea numerelor reale ecuația  $ax^4+bx^2+c=0$ , folosind o funcție pentru rezolvarea ecuației de gradul II.

36. Să se determine *punctele să* dintr-o matrice A cu m linii și n coloane folosind un subprogram care determină maximul dintr-un tablou liniar și o funcție care determină minimul dintr-un tablou liniar. (Observație : elementul  $a_{ij}$  se numește *element să* a matricei A dacă el este minim pe linia i și maxim pe coloana j sau invers).

37. Există numere care au proprietatea că se scriu, în două baze diferite, prin trei cifre identice. De exemplu, numărul  $273_{10}$  în baza 9 se scrie 333, și în baza 16 se scrie 111. Să se scrie un program care să determine toate numerele mai mici ca n care au această proprietate.

**38.** Se dau  $n$  mulțimi cu componente numere întregi, memorate într-o matrice cu  $2^n$  linii astfel: fiecare mulțime este dată prin numărul de elemente pe o linie a matricii iar elementele mulțimii pe linia următoare a matricei. Se cere să se afle elementele intersecției acestor mulțimi. Se va folosi o funcție care cauță un număr pe o linie a matricii, și o funcție care determină intersecția a două linii a matricei.

*Exemplu:* Pentru  $n=3$  și matricea

$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 8 & 1 & 4 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 \\ 7 & 8 & 1 & 3 & 4 \end{pmatrix}$$

se va afișa 1,4.

**39.** Scrieți un program care verifică relația

$$\text{cmmdc}(f_n, f_m) = f_{\text{cmmdc}(n,m)}$$

pentru două numere întregi  $m$  și  $n$  citite de la tastatură. Prin  $f_k$  s-a notat al  $k$ -lea element al șirului lui Fibonacci definit astfel:

$$f_0 = 1; \quad f_1 = 1; \quad f_n = f_{n-1} + f_{n-2}, \quad n \geq 2.$$

**40.** Se dă un polinom  $P$  de grad  $n$  și  $m$  valori reale  $a_1, a_2, \dots, a_m$ . Să se calculeze valoarea polinomului pentru cele  $m$  valori citite precum și maximul celor  $m$  valori  $P(a_i)$ .

**41.** Un centru meteo are  $n$  angajați. Fiecare angajat petrece la serviciu o anumită perioadă din zi, delimitată de momentele sosirii și plecării. În orice moment al zilei este prezent cel puțin un angajat. Să se determine perioada în care la centrul meteo se află un număr maxim de angajați.

**42.** Spunem despre un număr că are *proprietatea sufíxului*, dacă el apare ca sufíx al pătratului său, adică cifrele sale apar la sfârșitul șirului de cifre ce reprezintă acel număr ridicat la pătrat. De exemplu 9376 are proprietatea sufíxului în baza 10 ( $9376^2 = 87909376$ ). Este clar că baza în care lucrăm este importantă, un număr poate avea proprietatea sufíxului într-o bază și poate să

nu o aibă în altă bază. Scrieți un program care citește de la tastatură un număr natural n (scris în baza 10) și verifică dacă numărul are proprietatea sufixului în baza  $b \leq 10$ . Se va folosi o funcție de transformare a unui număr din baza 10 în baza b, o funcție care verifică dacă un număr are proprietatea sufixului într-o bază b.

**43.** Să se calculeze suma  $2^{x[1]} + 2^{x[2]} + \dots + 2^{x[n]}$ , pentru n număr natural și  $x = \{x[1], x[2], \dots, x[n]\}$  vector de numere întregi citite de la tastatură.

Observație : Valorile obținute pot fi foarte mari. De exemplu  $2^{100}$  conține 31 de cifre.

*Exemplu:* Pentru  $n = 5$  și  $x = \{5, 10, 35, 12, 22\}$  se obține suma 34363937824.

**44.** Se dau două numere a și b memorate ca siruri de caractere. Să se calculeze și afișeze suma și produsul celor două numere.

**45.** Se citesc de la tastatură n numere naturale cu cel mult 255 cifre. Să se afișeze numărul care are suma cifrelor cea mai mare. Se va folosi o funcție care are ca parametru un astfel de număr (memorat ca sir de caractere) și returnează suma cifrelor acestui parametru.

## 5.6. Teste grilă



5.6.1. Secvențele de mai jos au ca scop declararea a doi pointeri x și y către întregi. Care dintre ele nu sunt corecte?

a) `typedef int* intreg;`

`intreg x,y;`

c) `int *x,*y;`

e) `typedef intreg int;`

`intreg *x,*y;`

b) `typedef int intreg;`

`intreg *x,*y;`

d) `int &x,&y;`



5.6.2. Fie declarațiile de variabile:

`int a = 2, b, c = 5; int *x, *y;`

Precizați ce valori se vor afișa, în ordine, în urma execuției secvenței de program de mai jos.

a) 7, 7, 7

b) 7, 8, 9

c) 7, 8, 8

d) 7, 7, 8

e) 8, 8, 9

```
x = &c;  
a+= *x;  
cout<<a;  
b = ++a;  
y = &b ;  
cout<<*y;  
x = y;  
cout<<(*x)++;
```



5.6.3. Fie declarațiile de variabile:

`int a, b, c; int *x, *y, *z;`

Precizați ce valori se vor afișa, în ordine, în urma execuției secvenței de program de mai jos.

a) 510

b) 410

c) 400

d) 300

e) 301

```
a = 3;  
y = &c;  
*y = a++;  
z = &a;  
x = y;  
cout<<*x++;  
if (*x == y) cout<<1;  
else cout<<0;  
*z = *y;  
if (z == y) cout<<1;  
else cout<<0;
```



5.6.4. Fie un pointer x către un întreg. Care dintre instrucțiunile de mai jos realizează corect alocarea dinamică a memoriei?

- a) `x = (int)malloc(sizeof(int*));`
- b) `x = (int*)malloc(sizeof(int*));`
- c) `x = (int*)malloc(sizeof(int));`
- d) `*x = (int*)malloc(sizeof(int));`
- e) `*x = (int)malloc(sizeof(int*));`



5.6.5. Considerăm următoarea declarație de variabile:

`int *x,*y, z; float m;`

Care dintre secvențele de mai jos afișează corect media aritmetică a trei numere întregi citite de la tastatură?

a)  
`scanf("%d %d %d", x, y, z);  
m = (x + y + z) / 3;  
printf("%f", m);`

b)  
`scanf("%d %d %d", x, y, &z);  
m=(*x + *y + z) / 3;  
printf("%f", m);`

c)  
`scanf("%d %d %d", *x, *y, z);  
printf("%f", (*x+*y+z)/3);`

d)  
`cin>>*x>>*y>>z ;  
printf("%f", (*(x+y)+z)/2);`

e)  
`cin>>*x>>*y>>z;  
cout<<*x/3+*y/3+z/3;`



5.6.6. Programul următor își propune să afișeze jumătate din produsul a două numere citite, referite prin intermediul pointerilor a și b. Ce erori întâlnim în acest program ?

```

#include <stdio.h>
#include <stdlib.h>
int main ()
{
    int *a,*b;
    float x;
    a= (int*) malloc (sizeof (int));
    b= (int*) malloc (sizeof (int));
    scanf("%d %d", &a, &b);
    printf ("\n%f", x) ;
    free (a);
    free (b);
}

```

- a) Declarațiile de variabile sunt eronate
- b) Alocarea dinamică a memoriei este eronată
- c) Citirea de la tastatură nu se face corect
- d) Atribuirea nu calculează valoarea dorită
- e) Instrucțiunea de afișare a rezultatului este eronată



5.6.7. Fie următoarele declarații de variabile:

```
int **a, *b, c;
```

Care dintre expresiile de mai jos vor genera eroare la execuție?

- |                                   |                                |
|-----------------------------------|--------------------------------|
| a) <code>a=&amp;(&amp;c) ;</code> | b) <code>b=&amp;(**a) ;</code> |
| c) <code>*a=&amp;c;</code>        | d) <code>**a=&amp;b;</code>    |
| e) <code>*b=**a+c;</code>         |                                |



5.6.8. Pentru secvența de program următoare, care dintre afirmațiile de mai jos sunt adevărate?

```

#include <iostream.h>
int main ()
{
    int x,y;
    x=y=0;
    int &n=2 ;
    for (int &i=x;i<n;i++)
        cout<<y++;
}

```

- a) Linia `for` este eronată, deoarece contorul ciclului nu poate fi un pointer
- b) Instrucțiunea "`int &i=x`" este eronată
- c) Secvența este eronată, deoarece într-o instrucțiune `for` nu se poate scrie o declarație de sinonimie
- d) Secvența este corectă și va afișa valorile 0 și 1
- e) Secvența este corectă și va afișa valorile 1 și 2



5.6.9. Considerăm declarația: `int **p;`  
și atribuirea `p=&q;`

Alegeți declarația potrivită astfel, încât atribuirea să aibă sens.

- a) `int q;`
- b) `int *q;`
- c) `int ***q;`
- d) `int &q;`
- e) nici una.



5.6.10. Precizați valoarea variabilei `a` ca urmare a execuției programului următor:

- a) 1
- b) 97
- c) neprecizată
- d) o adresă
- e) programul este greșit

```
int main()
{
    int a;
    char b=1;
    a=*(int*)&b;
}
```



5.6.11. Precizați care dintre instrucțiunile de atribuire de mai jos face ca `x` să primească valoarea 0:

- a) I
- b) II
- c) ambele
- d) nici una
- e) programul este greșit

```
int main()
{
    int a=1, b=2;
    float x;
    x=a / * &b; // (I)
    x= (float) a/b; // (II)
}
```



5.6.12. Precizați valoarea pe care o va avea variabila a, ca urmare a execuției instrucțiunilor următoare:

```
int a, &b=a; b=1;
```

- a) nedefinită
- b) 0
- c) 1
- d) declarația este eronată
- e) dimensiunea tipului int



5.6.13. Care dintre instrucțiunile de tipărire vor afișa aceeași valoare:

- a) prima și a doua
- b) a doua și a treia
- c) a doua și a patra
- d) nici una
- e) programul este eronat

```
#include<stdio.h>
int main()
{
    int a=2, *p=&a;
    printf ("%d\n",*p+1);
    printf ("%d\n",*&p+1);
    printf ("%d\n", *(p+1));
    printf ("%d\n",*(p+1));
}
```



5.6.14. Care dintre tripletele de numere întregi date ca variante de răspuns pot fi introduse de la tastatură la execuția programului următor, astfel încât acesta să afișeze valoarea 1?

```
#include <stdio.h>
int main ()
{
    int a, b, c, m;
    const int x=0 ;
    scanf("%d %d %d" ,&a, &b, &c) ;
    m=a<b?a:b;
    printf ( "\n%d", x=m>c);
}
```

- a) 1, 2, 3
- b) 1, 3, 2
- c) 2, 3, 1
- d) 1,1, 1
- e) Instrucțiunea de afișare este eronată, din cauza atribuirii din funcția printf



5.6.15. În programul următor, care dintre cele patru instrucțiuni cout vor tipări valoarea 11 ?

```
#include <iostream.h>
int main()
{
    const int x=2,y=3;
    *(int*) &x=8;
    *(int*) &y=9;
    cout<<"\n"<<x+y;
    cout<<"\n"<<*(int*)&x+y;
    cout<<"\n"<<x+*(int*)y;
    cout<<"\n"<<*(int*)&x+*(int*)&y;
}
```

- a) prima
- b) a doua
- c) a treia
- d) a patra
- e) nici una

## 5.7. Răspunsuri la întrebările grilă

- 5.7.1. d) și e)
- 5.7.2. c)
- 5.7.3. d)
- 5.7.4. c)
- 5.7.5. b) și e)
- 5.7.6. c)
- 5.7.7. a) și d)
- 5.7.8. d)
- 5.7.9. b)
- 5.7.10. c)
- 5.7.11. a)
- 5.7.12. c)
- 5.7.13. d)
- 5.7.14. e)
- 5.7.15. b) și c)



# Fisiere

## 6.1. Tipuri de fișiere. Operații cu fișiere

**F**ișierul este o colecție de informații ordonată în înregistrări. Fișierele pot fi de mai multe tipuri, în funcție de datele ce se memorează în ele:

- **fișier text** conține doar caractere ASCII, iar înregistrările sunt separate prin caracterele speciale CR și LF
- **fișier standard** ( **fișierul header** ) - este un fișier text
- **fișierul binar** este un fișier ce conține date în format intern, iar înregistrările nu se mai separă prin caractere speciale

### 6.1.1. Operații cu fișiere

Prezentăm în continuare operațiile ce se pot efectua cu fișiere în limbajul C:

1. deschiderea unui fișier
2. crearea unui fișier
3. actualizarea unui fișier
4. consultarea unui fișier
5. poziționarea pe o anumită înregistrare din fișier
6. ștergerea unui fișier
7. ștergerea de înregistrări din fișier
8. redenumirea unui fișier

Se cunosc două nivele de prelucrare a fișierelor și anume:

1. **Nivelul inferior de prelucrare** – se face apel la funcțiile sistemului de operare MS-DOS
2. **Nivelul superior de prelucrare** – se folosesc funcții specializate în lucrul cu fișierele

*Observație:* În limbajul C++, nu există instrucțiuni de intrare/ieșire ci numai funcții specializate pentru aceste operații.

### 6.1.2. Prelucrarea fișierelor la nivel inferior

Pentru a putea utiliza prelucrarea fișierelor la nivel inferior, trebuie să includem în program următoarele fișiere standard : <io.h> și <fcntl.h>

- a) **Deschiderea unui fișier** se efectuează pentru ca să se asocieze fișierului de pe suportul fizic, un număr întreg numit descriptor de fișier = [df].

Descriptorii pentru fișierele standard sunt:



DENUMIRE FISIER	SEMNIFICATIE	VALOARE DESCRIPTOR
stdin	tastatura	0
stdout	monitor	1
stderr	fișier de erori	2
stdprn	imprimantă	3
stdaux	comunicație serială	4

Funcția utilizată este **open** și are următoarea sintaxă:



```
int open(const char *cale, int acces);
```

Unde: **cale** – este un pointer către o zonă de tip caracter, zonă în care este specificată calea de acces către fișierul ce trebuie deschis; Dacă descrierea completă a căii de acces lipsește, atunci fișierul va fi căutat în directorul curent.

**acces** - este o constantă predefinită care poate avea următoarele valori:



VALOARE	SEMNIFICATIE
O_RDONLY	deschis doar pentru citire
O_WRONLY	deschis doar pentru scriere
O_RDWR	deschis pentru citire și scriere
O_APPEND	deschis pentru adăugare la sfârșit
O_TEXT	deschidere fișier text
O_BINARY	deschiderea fișier binar

Se pot face și combinații între aceste valori: O\_RDONLY | O\_WRONLY, sau O\_RDONLY | O\_BINARY.

Exemplu: Următoarea secvență de program se poate folosi pentru a testa existența unui fișier cu numele "EXEMPLU.DAT":



```
int df;
if((df=open("C:\EXEMPLU.DAT",O_RDONLY)) == -1)
{
    printf("\nEroare la deschiderea fisierului");
    exit(1);
}
```

Valoarea ce o returnează funcția open poate fi:

- >0, dacă fișierul a fost găsit
- 1, dacă fișierul nu a fost găsit

b) Crearea unui fișier se efectuează cu ajutorul funcției **creat**, care are următoarea sintaxă:



```
int creat(const char *cale, int mod);
```

Unde: **cale** – este un pointer către o zonă de tip caracter, zonă în care este specificată calea de acces către fișierul ce trebuie deschis; Dacă descrierea completă a căii de acces lipsește, atunci fișierul va fi căutat în directorul curent.

**mod** – este modul de acces la fișierul respectiv:



<b>VALOARE</b>	<b>SEMNIFICATIE</b>
S_IREAD	creat pentru citire
S_IWRITE	creat pentru scriere
S_EXEC	fișier executabil

valorile returnate de funcția `creat` sunt:

>0, dacă fișierul poate fi creat

-1, în caz contrar

Exemplu: Următoarea secvență de program se poate folosi pentru a testa existența unui fișier cu numele “FISIER.DAT”:



```
int df;
if((df=creat("C:\FISIER.DAT",S_WRITE)) == -1)
{
    printf("\nEroare la deschiderea(crearea) fisierului");
    exit(1);
}
```

c) Citirea datelor dintr-un fișier se efectuează cu ajutorul funcției `read`, care are următoarea sintaxă:



```
int read(int df, void *buf, int lung);
```

Se citește secvențial din fișierul desemnat de descriptorul de fișier **df** un număr de octeți care se depun la adresa desemnată de **buf**.

### 6.1.3. Prelucrarea fișierelor la nivel superior

Operațiile permise asupra fișierelor și funcțiile asociate.

#### Entitatea *FILE*

Operațiile asociate fișierelor sunt:

- a) citire (*read*)
- b) scriere (*write*)
- c) adăugare (*append*)
- d) înlocuire/actualizare (*replace/update*)

Funcțiile pereche asociate acestor operații sunt:

- a) *fread()* sau *fopen()* cu argument *r/r+*
- b) *fopen()* cu argument *w/w+*
- c) *fwrite()* sau *fopen()* cu argument *a/a+*
- d) *fopen()* cu argument *w/w+* (operație similară oarecum, cu scrierea)

**De reținut!** Toate funcțiile ce urmează au prototip în *stdio.h*.

Toate informațiile despre un flux sunt stocate de către biblioteca standard într-o entitate de tip *FILE*. Această entitate este de fapt un tip utilizator creat folosind operatorul *typedef* împreună cu tipul de dată *struct*. Definiția lui apare în fișierul *stdio.h*.

#### Funcția *fopen()*

Crearea legăturii virtuale între program și entitatea fizică denumită fișier se face prin apelul funcției *fopen()*, care are definiția următoare în fișierul header *stdio.h*. Această funcție arată astfel:



**FILE\* fopen (const char \*filename, const char \*mode);**

Această funcție generează o adresă a unei structuri de tip *F/L/E*. Ceea ce înseamnă că ea pregătește operațiile tipice fișierelor. Acest canal de interacțiune devine valabil și rămâne astfel până în momentul închiderii lui explicite, cu funcția *fclose()*, sau până în momentul încheierii programului, situație în care, neexistând un apel la *fclose()*, este prevăzut un mod de tratare implicit din partea mediului de programare, anume închiderea automată prin apel de *fclose()* a tuturor fluxurilor (*stream*) rămase deschise.

Argumentele acestei funcții sunt:

- *filename* - arată calea către/de unde va fi salvat/există fișierul în cauză;
- *mode* - specifică varianta în care se fac prelucrările asupra fișierului, anume:
  - *r* = citire (*read*)
  - *w* = scriere (*write*)
  - *a* = adăugare (*append*) la sfârșitul fișierului; un fișier în mod *append* poate fi scris numai la sfârșitul său
  - *r+* = deschide fișier existent pentru actualizare (scriere și citire)
  - *w+* = creează fișier pentru actualizare
  - *a+* = deschidere pentru actualizarea la sfârșitul fișierului

Există un comportament predefinit al mediului de programare în ce privește lucrul asupra fișierelor. Astfel, dacă un fișier cu nume specificat nu există pe disc, atunci: dacă argumentul *mode* este *w* sau *a* (cu variantele *w+* și *a+*) el este nou creat, iar dacă argumentul *mode* este *r* se generează eroare (nu pot citi dintr-un fișier inexistent). Dacă fișierul există, atunci deschiderea cu *mode w* conduce la golirea sa (pierderea conținutului său).

Un fișier se poate *prelucra în modul text* dacă se adaugă sufixul *t* la variantele mode existente. De exemplu: *wt* sau *a+t* sau *at+t*, sau în mod binar dacă este adăugat, pe același principiu, sufixul *b* (de exemplu *rb*, *wb* sau *w+b* sau *ab+t*).

Dacă nu se specifică modul de deschidere, atunci compilatorul va lua decizia pe baza valorii variabilei globale *\_fmode* (cu valori *O\_BINARY* sau *O\_TEXT*) din headerul *fcntl.h*.

La fișierele *text* o succesiune de linii este separată prin caractere *newline* (*NL*) care pot fi convertite în perechea *CR-LF* (*carriage return* împreună cu *line feed*), în timp ce pentru fișierele binare nu se face această conversie.

Returnează un pointer *NULL* în caz de eroare (generată de diferite cauze) sau pointerul la *FILE* în caz de succes.

### Functia *fclose()*

Este funcția opusă lui *fopen()*. Rolul său este de a elibera canalul de comunicație existent (*fluxul*) către o anumită structură de tip *FILE*. Argumentul pe care o specificăm este pointerul la acea structură.

Sintaxa este preluată din fișierul *stdio.h*:



**FILE\* fopen (const char \*filename);**

Argumentul este deci numele fluxului deschis anterior cu *fopen()*. Este de menționat un amănunt tehnic. La orice deschidere de flux (*stream* sau canal de comunicație) se rezervă și o *zonă de memorie tampon* pentru acel flux. Închiderea unui flux are ca efect și golirea acelui tampon (cu termen de specialitate *flushing*), adică scrierea în fișier a datelor din memorie, odată cu eliberarea zonei de memorie astfel rezervate.

După cum se observă închiderea unui flux este importantă, având un dublu efect dorit: eliberarea zonei de memorie pentru alte operații, împreună cu scrierea pe disc a informațiilor rămase în memorie (*flushing of buffer*).

În caz de reușită funcția întoarce *0*. La eșec returnează *EOF* (*End Of File*).

### Functia *freopen()*

Cu ajutorul acestei funcții se schimbă asocierea unui flux existent. Astfel, dacă se deschid cu *fopen()* două fluxuri *A* și *B*, atunci prin apelul

*freopen()* se poate asocia fluxul *B* celui dedicat inițial lui *A*. Adică se atribuie un flux existent altui fișier. Înainte de schimbarea propriu-zisă, *freopen()* încearcă închiderea fișierului asociat în mod curent fluxului căruia îi va schimba asocierea. și în caz de reușită și în caz de eșec a încercării de închidere a fluxului căruia îi încearcă schimbarea asocierei, *freopen()* va deschide celălalt fișier.

Ca utilitate principală este *redirectionarea* fluxurilor standard *stdin* (tastatura), *stdout* (ecranul) și *stderr* (dispozitivul. standard de eroare) către un (alt) fișier.

Sintaxa este:



```
FILE* freopen (const char *numeFis, const char  
*mode, FILE *flux);
```

Noul nume de fișier este *numeFis*, modul de acces este dat prin *mode*, iar fluxul ce urmează a fi reatribuit este *flux*.

Dacă există vreo eroare, funcția întoarce un pointer *NULL*. La succes se întoarce pointerul către *flux*.

### Funcția *fgetc()*

Funcția *fgetc()* întoarce următorul caracter (octet) din fluxul de intrare specificat ca argument:



```
int fgetc(FILE flux);
```

Incrementează apoi indicatorul de poziție al fișierului. Caracterul este citit ca un *unsigned char*, convertit apoi la un întreg.

La întâlnirea sfârșitului de fișier funcția întoarce *EOF* (*End Of File*), la fel cu întâlnirea unei erori neașteptate.

### **Observație:**

În cazul fișierelor *binare*:

- deoarece *EOF* este codificat ca un întreg, iar un fișier binar este o mulțime de octeți, deci prin asimilare variabile de tip întreg cu semn, trebuie folosită funcția *feof()* pentru testarea sfârșitului de fișier
- pentru situațiile de eroare neașteptate, deoarece funcția întoarce tot *EOF* (ca și în cazul întâlnirii sfârșitului de fișier) trebuie folosită funcția *ferror()* pentru detectarea erorilor de fișier

**Observație:** În unele implementări apare și funcția *getc()*. Aceasta din urmă este identică cu *fgetc()*.

### **Funcția *fputc()***

Scrie caracterul preluat ca prim argument în fișierul al cărui flux este preluat drept al doilea argument. Incrementează apoi indicatorul de poziție în cadrul fișierului.



```
int fputc (int ch, FILE *flux);
```

Intern, funcția convertește caracterul preluat ca *int* la *unsigned char*. Funcția întoarce valoarea caracterului scris, în caz de succes și *EOF* în caz de eroare.

Trebuie făcută aceeași *observație*: pentru *fișierele binare*, *EOF* este un caracter valid, deci trebuie folosită funcția *ferror()* pentru detectarea eventualelor erori legate de fișier.

**Observație:** În unele implementări apare și funcția *putc()*. Aceasta din urmă este identică cu *fputc()*.

### **Functia fgets()**

Sintaxa este:



```
char *fgets (char *sir, int num, FILE *flux);
```

Această funcție preia maxim *num-1* caractere din fluxul de intrare *flux* și le salvează într-un vector de caractere indicat prin *sir*. După ultima scriere în vectorul de caractere se face automat adăugarea caracterului de sfârșit de sir ('*\0*).

În caz de reușită se întoarce pointerul la vectorul de caractere. În caz de eroare returnează un pointer *NULL*. Atunci când atinge sfârșitul fișierului această funcție întoarce tot un pointer *NULL*. Se impune aşadar folosirea funcțiilor de tratare a erorilor *ferror()* și *feof()*, pentru distingerea cazului real întâlnit.

### **Functia fputs()**

Sintaxa:



```
int fputs(const char *sir, FILE *flux);
```

Scrie în *flux* conținutul vectorului de caractere specificat prin *sir*. Valoarea de sfârșit de sir nu este scrisă în fișier.

La succes returnează valori non-negative. La eroare întoarce *EOF*. Pentru *fișiere de tip text* din cauza conversiilor amintite (*CR-LF* în *NL*) există posibilitatea ca numărul de caractere existent inițial în sir să nu corespundă numărului caracterelor efectiv scrise în fișier.

### **Functia fread()**

Efectul funcției este citirea unui număr specificat de obiecte de dimensiune stabilită tot de programator.

Sintaxa este următoarea:



```
size_t fread (void *buf, size_t octeti, size_t numar,  
FILE *flux);
```

Numărul obiectelor se specifică prin parametrul *numar*. Dimensiunea unui obiect, în octeți este dată prin parametrul *octeti*. Acestea sunt citite din fluxul *flux* și sunt stocate în zona tampon indicată prin *buf*. Ca și celelalte funcții de citire/scriere se modifică valoarea indicatorului de poziție, cu numărul de octeți citit.

Returnează numărul octetilor efectiv citiți. Dacă au fost returnate mai puține elemente decât cele citite efectiv este semn de eroare. Cauzele sunt fie necunoscute, fie s-a atins sfârșitul fișierului. De aceea este indicată utilizarea funcțiilor de eroare *feof()* sau *ferror()*.

Dacă se lucrează cu fișiere *text* pot apărea neconcordanțe între numărul de octeți cerut și cel efectiv întors, ceea ce se datorează conversiilor tipice anunțate (combinațiile *CR-LF* se convertesc în *NL - new line*).

### Funcția *fwrite()*

Complementara lui *fread()*. Aceasta scrie *numar* obiecte de dimensiune *octeti*. Elementele sunt preluate din zona tampon (buffer) indicată prin *buf*. Fluxul în care se scrie este *flux*.

Sintaxa este:



```
size_t fwrite(const void *buf, size_t octeit, size_t  
numar, FILE *flux);
```

Returnează numărul obiectelor scrise efectiv, care în caz de succes este egal cu numărul solicitat. În situația în care nu apare coincidență între ce se cere scris și ce s-a scris înseamnă apariția unei erori.

Pentru fișiere deschise în *mod text* deși apar conversiile amintite, identitatea trebuie să se mențină.

### **Funcția *fprintf()***

Scrie în fluxul specificat drept argument valorile argumentelor preluate ca parametri, pe baza indicațiilor din sirul de format.

Sintaxa:



```
int fprintf(FILE *flux, const char *format, ...);
```

În caz de succes, returnează numărul caracterelor efectiv afişate, iar în caz de eroare generează o *valoare negativă*.

Este o funcție cu număr variabil de argumente. Numărul maxim al acestora depinde de sistemul pe care este instalat compilatorul. În rest comportamentul și stilul de lucru sunt aceleași ca la clasica funcție *printf()*.

### **Funcția *fscanf()***

Are un comportament identic cu celebra *scanf()*, doar că citirea se face de la fluxul specificat.

Sintaxa este:



```
int fscanf (FILE *flux, const char *format, ...);
```

Cele trei puncte arată că funcția *fscanf()* (ca și *scanf()*) este o funcție cu număr variabil de argumente.

Returnează numărul argumentelor cărora li s-au atribuit efectiv valori. Apariția la ieșire lui *EOF* indică apariția unei erori înainte de efectuarea primei atribuirii.

## Funcții de tratare a erorilor

### Funcția *feof()*

Sintaxa acesteia este:



**int feof(FILE \*flux);**

Această funcție verifică poziția indicatorului de fișier. Dacă acest indicator de poziție a atins sfârșitul fișierului funcția returnează o valoare diferită de *0*. O valoare intermedie a indicatorului generează *zero*. Astfel că o parcurgere unui fișier (*binar*) s-ar putea face într-un ciclu, pe baza condiției:

```
...
FILE fisier = fopen("c:\\fisier.c", r);      // deschiderea fluxului asociat
fișierului fisier
if(fisier != NULL){
    while(!feof(fisier)) {           // not(0) = 1
        ... // instrucțiuni de prelucrare asupra fluxului fisier
    }
}
else { // tratarea erorii necunoscute
    printf("\n Eroare deschidere fișier!");
    printf("\n Cod eroare: %u", ferror(fisier)); // afișarea codului de eroare
    asociat
    exit(1);
// părâsirea absolută (imediată) a programului. prototip în stdlib.h:
// void exit (int) _ATTRIB_NORETURN;
}
...
...
```

Odată ajunși la sfârșitul unui fișier, eventualele operații de citire ulterioare vor returna *EOF* (în mod evident) până la apelarea funcției *rewind()* (cu semnificația de derulare la început) sau până în momentul în care indicatorului de poziție i se modifică valoarea, de exemplu prin una din funcțiile *fseek()* sau *fsetpos()*.

### **Observație:**

`feof()` este utilă mai ales în lucrul cu *fișiere binare*, în situația cărora valoarea *EOF* este și una validă (octet valid în cadrul fișierului, pe o poziție oarecare), pe lângă rolul caracterului de control sfârșit de fișier.

### **Funcția *ferror()***

Sintaxa este:



```
int ferror(FILE *flux);
```

Această eroare încearcă detectarea unei erori de fișier pentru fluxul preluat ca parametru. Dacă returnează zero atunci nu a apărut nici o eroare. O valoare diferită de zero indică o eroare.

Indicatorii de eroare sunt valabili până în momentul în care fluxul asociat fișierului este închis sau până în momentul în care se apelează una din funcțiile *rewind()* sau *clearerr()*. Pentru natura exactă a erorii se folosește funcția *perror()*.

### **Funcția *perror()***

Sintaxa:



```
void perror(const char *sir);
```

Realizează o corespondență biunivocă între valoarea variabilei globale *errno* și un fișier. Apoi scrie sirul *sir* la dispozitivul standard de eroare (*stderr*). Dacă valoarea lui *sir* nu este nulă atunci el este afișat, urmat de două puncte (.) și de mesajul de eroare tipic mediului de programare pentru eroarea apărută. De obicei *stderr* este 'deviat' către ecran (*stdout*). Sirul *sir* este dat de programator.

### Exemplul 1:

Lucrul cu fișierele - anume deschiderea unui flux asociat unui fișier - este studiată în programul urmator. Sunt exemplificate lucrul atât cu fișiere în *mod text* cât și în *mod binar*. Conținutul fișierului în studiu este în prealabil stabilit de programator. Pentru ca operațiile prezentate să aibă sens, atributul *read-only* al fișierului (în caz că este setat) trebuie dezactivat. Aceasta se face cu ajutorul comenziilor *Windows*: click dreapta pe numele fișierului, apoi alegerea comenzi *Properties*, după care de-bifarea opțiunii *Read-only* (din zona *Attributes*). Un atribut rămas pe *Read-only* generează erori ale mediului de programare.



```
// lucrul cu fișierele: deschiderea si inchiderea unui stream (flux)
#include<stdio.h>
#include<stdlib.h> // pentru functia exit()

int main(void){
    char ch, *sirEroare = "Eroare la deschidere fisier";
    int nrChar = 0; // contor al caracterelor
    FILE *pFisier; // pointerul la fisier; numele fluxului
    pFisier = fopen("c:\\test.txt", "W");
    // deschidere pentru scriere. Daca fisierul nu exista el este creat
    // Este util sa fie deschis in mod 'append' (adaugare la sfarsit),
    // pentru ca fisierul sa-si pastreze continutul intre doua deschideri.
    // Un caracter Enter la intrare se traduce in perechea CR-LF la
    // scriere in fisier.
    if (pFisier == NULL){ // eroare la deschidere fisier
        if(ferror(pFisier)) {
            perror(sirEroare);
            exit(1); // iesire din program
        }
    }
    else { while((ch = getchar()) != 'q'){
        fputc((int)ch, pFisier);
        nrChar++;
    }
}

printf("\nNumarul caracterelor scrise: %i", nrChar);
fclose(pFisier); // inchiderea fluxului asociat fisierului 'test.txt'

int i;
scanf("%i", &i);
}
```

## Exemplul 2:

Închiderea unui *stream* este analizată în programul urmator. Se observă situațiile în care *fclose()* este prezentă și absentă. Pe linia 38 există un comentariu care maschează execuția funcției *fclose()*. Renunțați la acest comentariu și reluați programul. Veți observa că nu este marcată nici o eroare. Avantajele închiderii explicite a unui flux: eliberarea zonei de memorie pentru alte operații și scrierea pe disc a informațiilor rămase în memorie (*flushing of buffer*).



```
// lucrul cu fisierele: deschiderea si inchiderea unui stream (flux)
#include<stdio.h>
#include<conio.h>
#include<stdlib.h> // pentru functia exit()

int main(void){
    char ch, *sirEroare = "Eroare la deschidere fisier";
    int nrChar = 0; // contor al caracterelor
    FILE *pFisier; // pointerul la fisier; numele fluxului

    pFisier = fopen("c:\\test1.txt", "w+");
    // Deschidere pentru scriere si actualizare.
    // Un caracter Enter la intrare se traduce in perechea CR-LF la
    // scriere in fisier.

    if (pFisier == NULL){ // eroare la deschidere fisier
        if(ferror(pFisier)) {
            perror(sirEroare);
            exit(1); // iesire din program
        }
    }
    else { while((ch = getchar()) != 'q'){
        fputc((int)ch, pFisier);
        nrChar++;
    }
}

printf("\nNumarul caracterelor scrise: %i", nrChar);

// apar erori in acest moment?
if(ferror(pFisier)) perror(sirEroare);
// fclose(pFisier); // inchiderea fluxului asociat fisierului 'test.txt'

getch();
}
```

### Exemplul 3:

Schimbarea asocierii unui flux existent este exemplificată în programul următor. Unul dintre fluxurile inițiale este atribuit celuilalt, după care se efectuează operații tipice. În final ambele sunt închise.

Rulați programul de cel puțin două ori. Remarcați pe *HDD* cum fișierul *test1.txt* conține mereu textul cel mai recent, în timp ce fișierul *test2.txt* toate celelalte texte date de utilizator la fiecare rulare. Atenție la modul de deschidere din fiecare caz al fluxurilor *pFis1* și *pFis2*. Rețineți aceste aspecte.

Creați două noi fluxuri *flux1* și *flux2*. Interschimbați-le asocierea. Citiți din *flux1* și scrieți în *flux2*. Închideți apoi ambele fluxuri fișier.



```
// lucrul cu fisierele: schimbarea asocierii unui flux fisier - freopen()
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>           // pentru functia exit()

int main(void){
    char ch ;
    char *sirErr1 = "Eroare la deschidere fisier 1" ;
    char *sirErr2 = "Eroare la deschidere fisier 2" ;
    int nrCh1 = 0, nrCh2 = 0 ; // contoare ale caracterelor (octetilor)
    int contor = 20; // numarul de executii al celei de-a doua bucle
    while()

        FILE *pFis1, *pFis2 ;      // pointerii la fisiere; numele fluxurilor
        // deschiderea fluxurilor și testele de eroare aferente.
        pFis1 = fopen("c:\\test1.txt", "wt+");
        pFis2 = fopen("c:\\test2.txt", "at+");
        // Un caracter Enter la intrare se traduce in perechea CR-LF la
        scriere in fisier.

        if (pFis1 == NULL){        // eroare la deschidere fisier1
            if(ferror(pFis1)) {
                perror(sirErr1) ;
                exit(1);           // iesire din program
            }
        }

        if (pFis2 == NULL){        // eroare la deschidere fisier2
            if(ferror(pFis2)) {
                perror(sirErr2) ;
                exit(1);           // iesire din program
            }
        }

    }
```

```

// scriere in fisierul 1
printf("\n Se scrie in fisierul test1.txt... \n");
while((ch = getchar()) != 'q'){
    fputc((int)ch, pFis1);
    nrCh1++;
}
printf("\nNumarul caracterelor scrise in fisierul >> test1.txt <<:
%i", nrCh1);
// citire din fisierul 2. Acesta este un fisier de proba existent apriori
// la locatia anuntata. Trebuie sa contina cel putin 20 caractere (!).
printf("\n\n\n Se citeste din fisierul test2.txt ... \n");
while(contor){
    printf("%c", fgetc(pFis2));
    nrCh2++; // incrementare a numarului de caractere citite.
    contor--; // decrementare la fiecare caracter citit.
}
printf("\nNumarul caracterelor citite din fisierul >> test2.txt
<<: %i", nrCh2);

// Inchiderea fluxului 2. Schimbarea asocierii fluxului 1, cu fluxul 2.
fclose(pFis2);
pFis2 = freopen("c:\\test2.txt", "at+", pFis1);
// asociiez pFis1 fisierului 'test2.txt'. Acesta nu mai este asociat
// fisierului test1.txt (!)
// scriu in fisierul 2, in mod adaugare, dupa schimbarea atribuirii.
nrCh1 = 0; // reset pe contorul de caractere pe care l-am stabilit,
// conventional, fisierului 'test1.txt'
printf("\n\n\n Se scrie in fisierul test2.txt, dupa reatribuirea
fluxurilor. \n");
while((ch = getchar()) != 'q'){
    fputc((int)ch, pFis2);
    nrCh1++;
}
printf("\nNumarul caracterelor scrise in fisierul >> test2.txt <<
dupa \ reatribuire: %i", nrCh1);

// inchiderea definitiva a fluxurilor
fclose(pFis1); fclose(pFis2);
getch();
}

```

#### Exemplul 4:

În programul următor se studiază comportamentul funcție *fgetc()* pentru un *stream* creat de programator. Citirea caracterelor se face până la întâlnirea sfărșitului de fișier (*EOF*) atât pentru *tip text* cât și pentru *tip binar*.

Observați rezultatele finale prin citirea fișierelor rezultate. Marcați concluziile pe care le considerați utile.



```
// lucrul cu fisierele: citirea dintr-un fisier cu ajutorul functiei fgetc()
// - text și binar
#include<stdio.h>
#include<conio.h>
#include<stdlib.h> // pentru functia exit()

int main(void){
    char ch, *sirEroare = "Eroare la deschidere fisier";
    int nrCharT = 32, nrCharB = 32; // numarul caracterelor ce vor
    fi citite
    int contorT = 0, contorB = 0; // contoare de citire pentru
    fiecare dintre fluxuri
    FILE *pFisT, *pFisB; // pointerii la fisier;
    // numele fluxurilor: text si binar

    pFisT = fopen("c:\\\\test1.txt", "rt");
    // deoarece urmeaza o citire din acest fisier se presupune ca el exista
    // deja pe HDD, la locatia anuntata. Altfel sunt generate erori de
    // compilator. Un caracter Enter la intrare se traduce in perechea CR-
    // LF la scriere in fisier.
    pFisB = fopen("c:\\\\test2.txt", "rb"); // citire in mod binar

    if (pFisT == NULL){ // eroare la deschidere fisier?
        if(ferror(pFisT)) {
            perror(sirEroare);
            exit(1); // iesire din program
        }
    }
    if (pFisT == NULL){ // eroare la deschidere fisier?
        if(ferror(pFisB)) {
            perror(sirEroare);
            exit(1); // iesire din program
        }
    }
}
```

```

// urmeaza operatiile specifice fisierelor
// pentru fisierul text
printf("\nCitire din fisierul test1.txt ...") ;

while(nrCharT){
    if(ferror(pFisT)) perror(sirEroare) ;
    printf("%c", fgetc(pFisT)) ;
    // citesc din fisierul test1.txt, prin
    // intermediul fluxului asociat pointerului 'pFisier'
    nrCharT-- ; // cat mai raman in bucla?
    contorT++ ; // cate caractere am scris pana acum?
}
printf("\n\nNumarul caracterelor citite: %i", contorT) ;

// pentru fisierul binar
printf("\n\nCitire din fisierul test2.txt ...") ;

while(nrCharB){
    if(ferror(pFisB)) perror(sirEroare) ;
    printf("%c", fgetc(pFisB)) ;
    // citesc din fisierul test1.txt, prin
    // intermediul fluxului asociat pointerului 'pFisier'
    nrCharB-- ; // cat mai raman in bucla?
    contorB++ ; // cate caractere am scris pana acum?
}
printf("\n\nNumarul caracterelor citite: %i", contorB) ;

// inchiderea celor doua fluxuri
fclose(pFisT) ; // inchiderea fluxului asociat fisierului 'test1.txt'
fclose(pFisB) ; // inchiderea fluxului asociat fisierului 'test2.txt'

puts("\n\nIncheiate apasand o tasta alfa-numerica...") ;
getch() ;
}

```

### Exemplul 5:

Programul următor arată comportamentul funcției complementare lui *fgetc()*, anume *fputc()*. Fișierul trebuie deschis în *mod actualizare* pentru ca o asemenea operație să poată fi valabilă. Se observă și valorile indicatorului de poziție în fișier, înainte și după scrierea caracterelor în fișierul respectiv. Prin

citirea fișierelor create pe *HDD* se poate observa cum caracterul '*\n*' fin sirul de scris este interpretat în mod diferit în funcție de natura fișierului: *text* sau *binar*.

Pentru ca operațiile prezentate să aibă sens atributul *read-only* al fișierului în caz că există, trebuie înlăturat. Aceasta se face cu ajutorul comenziilor *Windows*: click dreapta pe numele fișierului, apoi alegerea comenzi *Properties*, după care de-bifarea opțiunii *Read-only*. În final *Apply* și *OK* (sau direct *OK*).



```
// lucrul cu fisierele: scrierea intr-un fisier cu ajutorul functiei fputc()
// text si binar
#include<stdio.h>
#include<conio.h>           // pentru functia getch()
#include<stdlib.h>           // pentru functia exit()
#include<math.h>              // pentru functia pow()

typedef unsigned long int ULI ;

int main(void){
    char *sirEroare = "Eroare la deschidere fisier" ;
    char *sirDeScris = " Un mesaj de proba\n Pe doua randuri" ;
    char *pCh = sirDeScris ; // pointer la inceputul sirului de scris
    FILE *pFisT, *pFisB ;    // pointerii la fisier;
                            // numele fluxurilor: text si binar

    pFisT = fopen("c:\\test1.txt", "wt+"); // actualizare in mod
                                         // text.
                                         // Un caracter Enter la intrare se traduce in perechea CR-LF la
                                         // scriere in fisier.
    pFisB = fopen("c:\\test2.txt", "wb+"); // actualizare in mod
                                         // binar.

    if (pFisT == NULL){ // eroare la deschidere fisier?
        if(ferror(pFisT)) {
            perror(sirEroare) ;
            exit(1);           // iesire din program
        }
    }

    if (pFisB == NULL){ // eroare la deschidere fisier?
        if(ferror(pFisB)) {
            perror(sirEroare) ;
            exit(1);           // iesire din program
        }
    }
}
```

```

else { // incep scrierea in fisiere
    ULI i; // variabila locala corpului de instructiuni 'else'
    ULI contor = 0;
    printf("\n Se scrie in fisiere...\\n");
    while(*pCh++ != '\0') {
        i = (ULI) pow(2, 25);
        // numar natural posibil pentru acest tip de data (ULI)
        // 2^25 = 32 Mega
        fputc(*pCh, pFisT);
        fputc(*pCh, pFisB);

        printf(" %i", contor);
        contor++;
        for(; i>0; i--) ;// delay
    }
}

// inchiderea fluxurilor
fclose(pFisT);
fclose(pFisB);

printf("\n\n Dati un caracter alfa-numeric... ");
getch();
}

```

### Exemplul 6:

Preluarea unui sir de caractere dintr-un fisier (funcția *fgets()*) și observațiile asupra indicatorului de poziție al fișierului sunt urmărite în programul următor. Trebuie avută în vedere conversia posibilă din situația fișierelor de *tip text*. De aceea sirul destinație trebuie declarat de o dimensiune acoperitoare. Dacă la fiecare linie există un caracter de control *NL*, acesta va fi convertit la citire în două caractere de control: *CR-LF*. Puteți calcula astfel, aproximativ, numărul de caractere al sirului destinație.

Prin observarea rezultatului final se concluzionează: un sir dintr-un fisier (*text* sau *binar*) ce conține un caracter *new-line* este citit până la întâlnirea acelui caracter de control. Nu se trece mai departe, pe linia următoare pentru citirea restului sirului. Modificați conținutul fișierelor și observați comportamentul programului, prin urmărirea în final a conținutului celor două fișiere.



```
// lucrul cu fisierele: citirea intr-un fisier cu ajutorul functiei fgets() -  
// text si binar  
#include<stdio.h>  
#include<conio.h> // pentru functia getch()  
#include<stdlib.h> // pentru functia exit()  
typedef unsigned long int ULI; // tip de date 'intreg lung fara semn'  
                                // poate contine valori in gama [0 232-1]  
int main(void)  
{  
    char *sirEroare = "Eroare la deschidere fisier";  
    char sirCititT[32], sirCititB[32]; // sirurile citite sunt copiate in  
    // acestei vectori de caractere (text, respectiv binar)  
    const int nrMax = 30; // numarul maxim de caractere ce urmeaza  
    // sa fie citit  
    FILE *pFisT, *pFisB; // pointerii la fisier;  
                            // numele fluxurilor: text si binar  
    pFisT = fopen("c:\\test1.txt", "rt+"); // citire in mod text.  
    // Un caracter Enter la intrare se traduce in perechea CR-LF la  
    // scriere in fisier  
    pFisB = fopen("c:\\test2.txt", "rb+"); // citire in mod binar  
  
if (pFisT == NULL){ // eroare la deschidere fisier?  
    if(ferror(pFisT)) {  
        perror(sirEroare);  
        exit(1); // iesire din program  
    }  
}  
  
if (pFisB == NULL){ // eroare la deschidere fisier?  
    if(ferror(pFisB)) {  
        perror(sirEroare);  
        exit(1); // iesire din program  
    }  
    else { // incep citirea din fisiere  
        printf("\n S-au citit din fisiere urmatoarele siruri...\n");  
        puts(fgets(sirCititT, nrMax, pFisT));  
        puts(fgets(sirCititB, nrMax, pFisB));  
  
    } // end_else  
    // afisarea continutului sirurilor buffer  
    printf("\n\n s Ce s-a scris in sirul sirCititT: \n %s", sirCititT);  
    printf("\n Iar ce s-a scris in sirul sirCititB: \n %s", sirCititB);  
    // inchiderea fluxurilor  
    fclose(pFisT);  
    fclose(pFisB);  
    printf("\n\n Dati un caracter alfa-numeric... ");  
    getch();  
}
```

### Exemplul 7:

Funcția complementară lui *fgets()* este *fputs()*. Utilizarea acesteia este studiată în programul următor. Ca și în cazul lui *fgets()* indicatorul de poziție în fișier este modificat în consecință.

Observați diferența între numărul de caractere scris efectiv și lungimea reală a celor două șiruri ce au fost scrise.



```
// lucrul cu fisierele: scrierea intr-un fisier cu ajutorul functiei fputs()  
// urmarirea indicatorilor de pozitie inainte si dupa scriere - text si  
binar  
#include<stdio.h>  
#include<conio.h> // pentru functia getch()  
#include<stdlib.h> // pentru functia exit()  
#include<string.h> // pentru functia strlen()  
  
typedef unsigned long int ULI ;  
  
int main(void){  
    char *sirEroare = "Eroare la operatie asupra fisier" ;  
    char *sirDeScrisT = "\n Un mesaj de proba, \n in mod text" ;  
    char *sirDeScrisB = "\n Un mesaj de proba, \n in mod binar" ;  
  
    FILE *pFisT, *pFisB ; // pointerii la fisier;  
                           // numele fluxurilor: text si binar  
  
    // deschiderea fluxurilor, impreuna cu testelete de eroare aferente  
    pFisT = fopen("c:\\test1.txt", "wt+" );  
    // actualizare prin stergerea continutului anterior, in mod text.  
    // Un caracter Enter la intrare se traduce in perechea CR-LF la  
    // scriere in fisier.  
    pFisB = fopen("c:\\test2.txt", "wb+" );  
    // actualizare prin stergerea continutului anterior, in mod binar.  
  
    if (pFisT == NULL){ // eroare la deschidere fisier?  
        if(ferror(pFisT)) {  
            perror(sirEroare) ;  
            exit(1) ; // iesire din program  
        }  
    }  
  
    if (pFisB == NULL){ // eroare la deschidere fisier?  
        if(ferror(pFisB)) {  
            perror(sirEroare) ;  
            exit(1) ; // iesire din program  
        }  
    }
```

```

// operatia efectiva asupra fisierelor
else {           // nu este eroare => incep scrierea in fisiere.
    ULI lenT = strlen(sirDeScrisT), lenB = strlen(sirDeScrisB) ;
                    // lungimile celor doua siruri de scris.
    fpos_t pozT, pozB ;

    printf("\n Lungimile celor doua siruri ce urmeaza sa fie scrise:
\text = %i, binar = %i", lenT, lenB) ;

    // test eroare indicator fisier text: inainte de scriere
    if(!fgetpos(pFisT, &pozT))
        printf("\n Indicatorul de pozitie al fisierului text
(inainte): %u", pozT) ;
    else {
        if(ferror(pFisT)) {
            perror(sirEroare) ;
            exit(1) ;           // iesire din program
        }
    }

    // test eroare indicator fisier binar: inainte de scriere
    if(!fgetpos(pFisB, &pozB))
        printf("\n Indicatorul de pozitie al fisierului binar
(inainte): %u", pozB) ;
    else {
        if(ferror(pFisB)) {
            perror(sirEroare) ;
            exit(1) ;           // iesire din program
        }
    }

    // scrierea efectiva. Teste de eroare la scriere (fputs() intoarce valori
    // non-negative in caz de succes. Intoarce EOF in caz contrar)
    printf("\n\n Se scrie in fisiere...\n") ;

    if(fputs(sirDeScrisT, pFisT) == EOF) {
        if(ferror(pFisT)) {
            perror(sirEroare) ;
            exit(1) ;           // iesire din program
        }
    }

    if(fputs(sirDeScrisB, pFisB) == EOF) {
        if(ferror(pFisB)) {
            perror(sirEroare) ;
            exit(1) ;           // iesire din program
        }
    }
}

```

```

// test eroare indicator fisier text: dupa scriere
if(!fgetpos(pFisT, &pozT))
    printf("\n Indicatorul de pozitie al fisierului text (dupa):
%u", pozT) ;
else {
    if(ferror(pFisT)) {
        perror(sirEroare) ;
        exit(1) ;           // iesire din program
    }
}

// test eroare indicator fisier binar: dupa scriere
if(!fgetpos(pFisB, &pozB))
    printf("\n Indicatorul de pozitie al fisierului binar
(dupa): %u", pozB) ;
else {
    if(ferror(pFisB)) {
        perror(sirEroare) ;
        exit(1) ;           // iesire din program
    }
}
} // end_else

// inchiderea fluxurilor
fclose(pFisT) ;
fclose(pFisB) ;

// sfarsit
printf("\n\n Dati un caracter alfa-numeric... ");
getch() ;
}

```

### Exemplul 8:

Funcția *fread()* este urmărită în programul următor. Atenție aici la primul argument - *void \*buf*. Trebuie să existe operatorul de conversie explicită, către tipul de dată pe care doriți să-l citiți. După cum se poate concluziona, tipul *void\** reprezintă o generalizare, adică 'pointer către orice'. Trebuie remarcate și conversiile între caracterele de control *NL* și *CR-LF*.



```
// lucrul cu fisierele: scrierea intr-un fisier cu ajutorul functiei  
fwrite()  
// urmarirea indicatorilor de pozitie inainte si dupa scriere. - text si  
binar  
#include<stdio.h>  
#include<conio.h> // pentru functia getch()  
#include<stdlib.h> // pentru functia exit()  
#include<string.h> // pentru functia strlen()  
  
typedef unsigned long int ULI ;  
  
int main(void){  
    char *sirEroare = "Eroare la operatie asupra fisier" ;  
    char *sirDeScrisT = " Un mesaj de proba, in mod text." ;  
    char *sirDeScrisB = " Un mesaj de proba, in mod binar." ;  
  
    const int nrObiecte = 8 ; // voi scrie/citi 8 obiecte ...  
    const int dimObiect = 2 ; // ... de lungime 2 octeti, fiecare.  
  
    FILE *pFisT, *pFisB ; // pointerii la fisier;  
                           // numele fluxurilor: text si binar  
  
    // deschiderea fluxurilor, impreuna cu testele de eroare aferente  
    pFisT = fopen("c:\\test1.txt", "wt");  
    // actualizare prin stergerea continutului anterior, in mod text.  
    // Un caracter Enter la intrare se traduce in perechea CR-LF la  
    // scriere in fisier.  
    pFisB = fopen("c:\\test2.txt", "wb+");  
    // actualizare prin stergerea continutului anterior, in mod binar.  
  
    if (pFisT == NULL){ // eroare la deschidere fisier?  
        if(ferror(pFisT)) {  
            perror(sirEroare) ;  
            exit(1) ; // iesire din program  
        }  
    }  
  
    if (pFisB == NULL){ // eroare la deschidere fisier?  
        if(ferror(pFisB)) {  
            perror(sirEroare) ;  
            exit(1) ; // iesire din program  
        }  
    }
```

```

// operatia efectiva asupra fisierelor
else {           // nu este eroare => incep scrierea in fisiere.
    ULI lenT = strlen(sirDeScrisT), lenB = strlen(sirDeScrisB) ;
    // lungimile celor doua siruri de scris.
    fpos_t pozT, pozB ;

    printf("\n Lungimile celor doua siruri din care urmeaza sa se
scrise: \ text = %i, binar = %i", lenT, lenB) ;

    // test eroare indicator fisier text: inainte de scriere
    if(!fgetpos(pFisT, &pozT))
        printf("\n Indicatorul de pozitie al fisierului text
(inainte): %u", pozT) ;
    else {
        if(ferror(pFisT)) {
            perror(sirEroare) ;
            exit(1) ;           // iesire din program
        }
    }

    // test eroare indicator fisier binar: inainte de scriere
    if(!fgetpos(pFisB, &pozB))
        printf("\n Indicatorul de pozitie al fisierului binar
(inainte): %u", pozB) ;
    else {
        if(ferror(pFisB)) {
            perror(sirEroare) ;
            exit(1) ;           // iesire din program
        }
    }

    // scrierea efectiva, impreuna cu testele de eroare aferente legate de
    functia folosita.
    {
        ULI scriseT, scriseB ;
        // numarul obiectelor scrise efectiv - raportat de functia fwrite()
        printf("\n\n Se scrie in fisiere...\n") ;
        if((scriseT = fwrite((char*)sirDeScrisT, dimObiect, nrObiecte,
pFisT)) == nrObiecte)
            printf("\n Succes! Numarul efectiv de caractere scrise
(text): %u", scrise T*dimObiect) ;

        else {
            if(ferror(pFisT)) {
                perror(sirEroare) ;
                exit(1) ;           // iesire din program
            }
        }
    }
}

```

```

// aceleasi operatii pentru fisierul binar
if((scriseB = fwrite((char*)sirDeScrisB, dimObiect, nrObiecte,
pFisB)) == nrObiecte)
printf("\n Succes! Numarul efectiv de caractere scris (binar):
%u", scrise B*dimObiect);

else {
    if(ferror(pFisB)) {
        perror(sirEroare);
        exit(1); // iesire din program
    }
}

// test eroare indicator fisier text: dupa scriere
if(!fgetpos(pFisT, &pozT))
printf("\n\n Indicatorul de pozitie al fisierului text
(dupa): %u", pozT);
else {
    if(ferror(pFisT)) {
        perror(sirEroare);
        exit(1); // iesire din program
    }
}

// test eroare indicator fisier binar: dupa scriere
if(!fgetpos(pFisB, &pozB))
printf("\n Indicatorul de pozitie al fisierului binar
(dupa): %u", pozB);
else {
    if(ferror(pFisB)) {
        perror(sirEroare);
        exit(1); // iesire din program
    }
}
} // end_else

} // end_scriereEfectiva
// inchiderea fluxurilor
fclose(pFisT);
fclose(pFisB);

// sfarsit
printf("\n\n Dati un caracter alfa-numeric... ");
getch();
}

```

### Exemplul 9:

Complementara lui *fread()* este *fwrite()*. Studiul său se face în programul următor. Efectele sunt aceleași ca la orice funcție de citire scriere: se modifică indicatorul de poziție în fișier, iar conversiile de caractere de control sunt inevitabile pentru fișierele de *tip text*.



```
// lucrul cu fisierele: citirea dintr-un fisier cu ajutorul functiei fread()
// urmarirea indicatorilor de pozitie inainte si dupa scriere. - text si
binar
#include<stdio.h>
#include<conio.h> // pentru functia getch()
#include<stdlib.h> // pentru functia exit()
#include<string.h> // pentru functia strlen()
typedef unsigned long int ULI ;
void finalSir(const char*, ULI*, const ULI) ;

int main(void){
    char *sirEroare = "Eroare la operatie asupra fisier" ;
    void *sirCititT[32], *sirCititB[32] ;
    // cele doua zone tampon (buffer) in care
    // se salveaza textul citit din fisiere (text respectiv binar).

    const int nrObiecte = 8 ; // voi citi 8 obiecte ...
    const int dimObiect = 3 ; // ... de lungime 2 octeti, fiecare.
    FILE *pFisT, *pFisB ; // pointerii la fisier;
                           // numele fluxurilor: text si binar
    // deschiderea fluxurilor, impreuna cu testele de eroare aferente
    pFisT = fopen("c:\\test1.txt", "rt") ;
    // mod citire si actualizare, in mod text.
    // Un caracter Enter la intrare se traduce in perechea CR-LF la
    scriere in fisier.
    pFisB = fopen("c:\\test2.txt", "rb") ;
    // mod citire si actualizare, in mod binar.

    if (pFisT == NULL){ // eroare la deschidere fisier?
        if(ferror(pFisT)) {
            perror(sirEroare) ;
            exit(1) ; // iesire din program
        }
    }
    if (pFisB == NULL){ // eroare la deschidere fisier?
        if(ferror(pFisB)) {
            perror(sirEroare) ;
            exit(1) ; // iesire din program
        }
    }
}
```

```

// operatia efectiva asupra fisierelor
else {           // nu este eroare => incep citirea din fisiere.
    fpos_t pozT, pozB ; // indicatorii de pozitie ai celor doua tipuri de
                          fisiere.
    // test eroare indicator fisier text: inainte de scriere
    if(!fgetpos(pFisT, &pozT))
        printf("\n Indicatorul de pozitie al fisierului text
(inainte): %u", pozT) ;
    else {
        if(ferror(pFisT)) {
            perror(sirEroare) ;
            exit(1) ;           // iesire din program
        }
    }

    // test eroare indicator fisier binar: inainte de scriere
    if(!fgetpos(pFisB, &pozB))
        printf("\n Indicatorul de pozitie al fisierului binar
(inainte): %u", pozB) ;
    else {
        if(ferror(pFisB)) {
            perror(sirEroare) ;
            exit(1) ;           // iesire din program
        }
    }

    // citirea efectiva, impreuna cu testele de eroare aferente legate de
    functia folosita.
    {      // un nou corp de instructiuni
        ULI cititeT, cititeB ;       // numarul obiectelor citite efectiv -
        raportat de functia fread()
        char *pVar ;
        int contor ;

        printf("\n\n Se citeste din fisiere...\n") ;

        if((cititeT = fread((char*)sirCititT, dimObiect, nrObiecte,
pFisT)) == nrObiecte)
        {
            printf("\n Succes! Numarul efectiv de caractere citite
(text): %u",

            cititeT*dimObiect) ;
            finalSir((const char*)sirCititT, &cititeT, dimObiect) ;
            printf("\n Ceea ce s-a citit (text): %s", (char*)sirCititT) ;
        }
    }
}

```

```

else {
    if(ferror(pFisT)) {
        perror(sirEroare) ;
        exit(1) ;           // iesire din program
    }
}

// aceleasi operatii pentru fisierul binar
if((cititeB = fread((char*)sirCititB, dimObiect, nrObiecte,
pFisB)) == nrObiecte)
{
    printf("\n\n Succes! Numarul efectiv de caractere citite
(binar): %u", cititeB*dimObiect) ;
    finalSir((const char*)sirCititB, &cititeB, dimObiect) ;
    printf("\n Ceea ce s-a citit (binar): %s", (char*)sirCititB) ;
}

else {
    if(ferror(pFisB)) {
        perror(sirEroare) ;
        exit(1) ;           // iesire din program
    }
}

// test eroare indicator fisier text: dupa scriere
if(!fgetpos(pFisT, &pozT))
    printf("\n\n Indicatorul de pozitie al fisierului text
(dupa): %u", pozT) ;
else {
    if(ferror(pFisT)) {
        perror(sirEroare) ;
        exit(1) ;           // iesire din program
    }
}

// test eroare indicator fisier binar: dupa scriere
if(!fgetpos(pFisB, &pozB))
    printf("\n Indicatorul de pozitie al fisierului binar
(dupa): %u", pozB) ;
else {
    if(ferror(pFisB)) {
        perror(sirEroare) ;
        exit(1) ;           // iesire din program
    }
}
} // end_else

} // end_citireEfectiva

```

```

// inchiderea fluxurilor
fclose(pFisT) ;
fclose(pFisB) ;

// sfarsit
printf("\n\n Dati un caracter alfa-numeric... ");
getch() ;
}

// implementarea functiei declarate local (functie proprie
programatorului).
// Se marcheaza explicit sfarsitul sirului tampon, deoarece fread() nu
face acest lucru automat.
void finalSir( const char *sirCitit,
/* text sau binar, nu conteaza din cauza apelului referinta*/
ULI *citite,
const ULI dimObiect) {
char *pVar ; // pointer variabil, pentru deplasarea in interiorul
// sirului preluat ca argument.
int contor; // variabila locala

// instructiunile proprii functiei
contor = *citite*dimObiect ;
pVar = (char*) sirCitit ;

while(contor--) pVar++ ;
// variabila contor se decrementeaza dupa ce se testeaza valoarea de
adevar a conditiei logice.
*pVar = '\0' ; // marchez explicit sfarsitul sirului
}

```

### Exemplul 10:

Funcțiile *fprintf()* și *fscanf()* apar în studiu în programul următor. După cum se cunoaște de la funcțiile consacrate *printf()* și *scanf()*, comportamentul celor dedicate fișierelor lor este similar. Observați și aici valoarea indicatorului de poziție. Condiția ca *fscanf()* să funcționeze corect este ca fișierul din care se face citirea acelor numere să conțină pe prima linie valori valide specifice tipului de dată pentru care se face citirea. Aici sunt citite 3 valori reale simplă precizie și 2 valori întregi scurt cu semn. Fișierul *test2.txt* trebuie să conțină pe prima linie valori valide.



```
// lucrul cu fisierele: citirea scrierea formatate intr-un/dintr-un fisier
// cu ajutorul functiilor pereche fprintf() / fscanf().
// urmarirea indicatorilor de pozitie inainte si dupa scriere. - text si
binar

#include<stdio.h>
#include<conio.h> // pentru functia getch()
#include<stdlib.h> // pentru functia exit()

typedef unsigned long int ULI ;

int main(void){
    char *sirEroare = "Eroare la operatie asupra fisier";
    char *sirFormat = "\n Se scriu %f, %f, %f drept valori reale si
%oi, %i ca valori intregi.";

    const int nrObiectReal = 3 ; // voi scrie 3 obiecte reale...
    const int nrObiectInt = 2 ; // ... si 2 obiecte intregi.

    FILE *pFisT, *pFisB ; // pointerii la fisier;
                           // numele fluxurilor: text si binar

    // deschiderea fluxurilor, impreuna cu testelete de eroare aferente
    pFisT = fopen("c:\\test1.txt", "wt+");
                           // actualizare prin adaugare, in mod text.
    // Un caracter Enter la intrare se traduce in perechea CR-LF la
    // scriere in fisier.
    pFisB = fopen("c:\\test2.txt", "at+");
                           // actualizare prin adaugare, in mod binar.

    if (pFisT == NULL){ // eroare la deschidere fisier?
        if(ferror(pFisT)) {
            perror(sirEroare);
            exit(1); // iesire din program
        }
    }

    if (pFisB == NULL){ // eroare la deschidere fisier?
        if(ferror(pFisB)) {
            perror(sirEroare);
            exit(1); // iesire din program
        }
    }
}
```

```

// operatia efectiva asupra fisierelor
else {           // nu este eroare => incep citirea din fisiere.
    fpos_t pozT, pozB ; // indicatorii de pozitie ai celor doua tipuri de
fisiere.

// test eroare indicator fisier text: inainte de scriere
if(!fgetpos(pFisT, &pozT) )
    printf("\n Indicatorul de pozitie al fisierului text
(inainte): %u", pozT) ;
else {
    if(ferror(pFisT)) {
        perror(sirEroare) ;
        exit(1) ;           // iesire din program
    }
}
// test eroare indicator fisier binar: inainte de scriere
if(!fgetpos(pFisB, &pozB) )
    printf("\n Indicatorul de pozitie al fisierului binar
(inainte): %u", pozB) ;
else {
    if(ferror(pFisB)) {
        perror(sirEroare) ;
        exit(1) ;           // iesire din program
    }
}

// operatiile efective, impreuna cu testelete de eroare legate de functiile
folosite.
{
    ULI scriseT, cititeB ;// numarul caaracterelor scrisen mod text, si
                           // citite in mod binar
    int i1, i2 ;          // variabilele intregi citite binar.
    float r1, r2, r3 ;    // variabilele reale citite binar.
    printf("\n\n Se scrie in fisierul text...") ;
    scriseT = fprintf(pFisT, sirFormat, 2.71828, -3.141592, -0.17, 3,
4) ;
    printf("\n Numarul efectiv de caractere scrise in fisier (!) text):
%u", scriseT) ;

// operatiile pentru fisierul binar
printf("\n\n Se citeste din fisierul binar...") ;

    cititeB = fscanf(pFisB, "%f,%f,%f,%i,%i", &r1, &r2, &r3,
&i1, &i2) ;
    printf("\n Numarul efectiv de obiecte atribuite(!) (binar): %u",
cititeB) ;
    printf("\n Valorile variabilelor dupa citirea din fisier: %f, %f,
%f si %i, %i", r1, r2, r3, i1, i2) ;

```

```

// test eroare indicator fisier text: dupa scriere
if(!fgetpos(pFisT, &pozT))
    printf("\n\n Indicatorul de pozitie al fisierului text
(dupa): %u", pozT) ;
else {
    if(ferror(pFisT)) {
        perror(sirEroare) ;
        exit(1);           // iesire din program
    }
}

// test eroare indicator fisier binar: dupa scriere
if(!fgetpos(pFisB, &pozB))
    printf("\n Indicatorul de pozitie al fisierului binar
(dupa): %u", pozB) ;
else {
    if(ferror(pFisB)) {
        perror(sirEroare) ;
        exit(1);           // iesire din program
    }
}
} // end_else
} // end_citireEfectiva

// inchiderea fluxurilor
fclose(pFisT) ;
fclose(pFisB) ;

// sfarsit
printf("\n\n Dati un caracter alfa-numeric... ");
getch() ;
}

```

### Exemplul 11:

Funcțiile de tratare a erorilor legate de fișiere sunt urmărite în programul următor. Erorile sunt provocate pentru a putea urmări diferitele aspecte ale funcționării acestor funcții. Fișierul fie nu există și se dorește citirea din acesta, fie are atributul *read-only* activ și se dorește scrierea, fie se cer mai multe caractere de citit decât există în mod real, depășindu-se dimensiunea acestuia (atingerea sfârșitului de fișier *EOF*).

Goliți fișierele *test1.txt* și *test2.txt* umplute în urma rulărilor multiple ale programului (sau programelor anterioare) și reluați următor. Observați în ce situații testul de depășire a indicatorului de poziție în fișier are condiția adevărată. Rețineți aspectele și situațiile practice întâlnite.

Modificați modul de acces la deschiderea fluxurilor și încercați operații opuse modului ales, din punct de vedere logic (de exemplu fișier deschis pentru *citire* în care se dorește *scrierea*). Observați cum operațiile aflate în contradicție cu modul de deschidere al fișierului sunt ignore (nu au efect la rulare).

Activați atributul *read-only* al fișierelor, apoi alegeti un mod de deschidere impropriu și observați comportamentul mediului integrat de programare. Rețineți situațiile practice de eroare.



```
// lucrul cu fisierele: functiile de tratare a erorilor

#include<stdio.h>
#include<conio.h> // pentru functia getch()
#include<stdlib.h> // pentru functia exit()

int main(void){
    char *sirEroareT = "\nLa operatia asupra fisierului text" ;
    char *sirEroareB = "\nLa operatia asupra fisierului binar" ;
    char bufferText[128], bufferBin[128] ;
    // numarul maxim de caractere (octeti) cititi poate
    // fi cel specificat de dimensiunea vectorilor.

    FILE *pFisT, *pFisB ; // pointerii la fisier;
                           // numele fluxurilor: text si binar

    // deschiderea fluxurilor, impreuna cu testelete de eroare aferente
    pFisT = fopen("c:\\test1.txt", "rt") ;
    // Deschis in mod citire, pentru tip text.
    // Ramane sa modificati pe rand aceste moduri de deschidere si sa
    // observati efectele, vizualizand de fiecare data continutul fisierelor.
    // Daca fisierul test1.txt contine un numar dat de caractere, atunci
    // incercarea de a citi mai multe caractere genereaza EOF.
    // Modificati modul de deschidere in "wt". Observati
    // comportamentul compilatorului, care se blocheaza.

    // pFisB = fopen("c:\\test2.txt", "wb") ;
    // pFisB = fopen("c:\\test2.txt", "ab") ;
    pFisB = fopen("c:\\test2.txt", "rb") ;
```

```

// Deschidere in mod adaugare, sau scriere sau citire. Ramane sa
// modificati pe rand aceste moduri de deschidere si sa observati
// efectele, vizualizand de fiecare data continutul fisierelor.
// Stabiliti pentru fisierul de pe HDD atributul read-only pe ON.
// Compilatorul va genera erori grave si se va blocha.
// Dezactivati apoi atributul read-only pentru a obtine o functionare
// normala a programului. Sau schimbati modul de deschidere al
// fisierului, astfel incat acesta sa fie compatibil cu atributul read-
// only.
// Daca modul de deschidere este 'append' si se doresc operatii de
// citire acestea nu se executa cu succes.

if (pFisT == NULL) { // eroare la deschidere fisier?
    if(ferror(pFisT)) {
        perror(sirEroareT) ;
        exit(1); // iesire din program
    }
}
if (pFisB == NULL) { // eroare la deschidere fisier?
    if(ferror(pFisB)) {
        perror(sirEroareB) ;
        exit(1); // iesire din program
    }
}
// operatia efectiva asupra fisierelor
else { // nu este eroare => incep operatiile pe fisiere.
    fpos_t pozT, pozB; // indicatorii de pozitie ai celor doua tipuri de
    fisiere.
    // test eroare indicator fisier text: inainte de scriere
    if!fgetpos(pFisT, &pozT)
        printf("\n Indicatorul de pozitie al fisierului text
(inainte): %u", pozT) ;
    else {
        if(ferror(pFisT)) {
            perror(sirEroareT) ;
            exit(1); // iesire din program
        }
    }
    // test eroare indicator fisier binar: inainte de scriere
    if!fgetpos(pFisB, &pozB)
        printf("\n Indicatorul de pozitie al fisierului binar
(inainte): %u", pozB) ;
    else {
        if(ferror(pFisB)) {
            perror(sirEroareB) ;
            exit(1); // iesire din program
        }
    }
}

```

```

// operatiile efective, impreuna cu testele de eroare legate de functiile
// folosite.
{
    int nrObiecte = 4; // numarul de obiecte ce urmeaza sa fie citite
    // (variabila locala)
    int dimObiect = 2; // numarul de octeti al unui obiect (variabila
    // locala)
    printf("\n\n Se citeste din fisierul text...") ;
    printf("\n S-au citit efectiv %u obiecte", fread(bufferText,
    dimObiect, nrObiecte, pFisT)) ;

    if(feof(pFisT)) perror(sirEroareT); // test de EOF.

    int i = 0 ;
    while(i < nrObiecte*dimObiect) i++ ;
    bufferText[i] = '\0'; // marcarea explicita a sfarsitului de sir.

    // puts(bufferText);
    printf("\n Buffer-ul contine sirul: %s", bufferText);

    // operatie improprie modului de deschidere ales
    fprintf(pFisT, "\n Operatie interzisa, conform modului de
    deschidere ales");
    // aceasta operatie este ignorata (!)
}

// operatiile pentru fisierul binar
{
    int nrObiecte = 24; // numarul de obiecte ce urmeaza sa fie citite
    // (variabila locala)
    int dimObiect = 4; // numarul de octeti al unui obiect (variabila
    // locala)

    printf("\n\n Se citeste din fisierul binar...") ;

    // o citire cu un numar de caractere peste dimensiunea fisierului,
    // considerand fisierul initializat cu mai putine caractere decat
    // valoarea nrObiecte*dimObiect din acest moment.
    // O rulare multipla a programului va schimba continutul fisierului
    // si deci si valoarea logic a conditiei din if(feof()) de mai jos.
    fread(bufferBin, dimObiect, nrObiecte, pFisB) ;
    if(feof(pFisB)) perror(sirEroareB); // test de EOF.
    int i = 0 ;
    while(i < nrObiecte*dimObiect) i++ ;
    bufferText[i] = '\0'; // marcarea explicita a sfarsitului de sir.

    printf("\n Buffer-ul contine sirul: %s", bufferBin);
}

```

```

// o scriere fara modificarea indicatorului de pozitie, dar cu
// schimbarea fluxului asociat
fclose(pFisT) ;
pFisT = freopen("c:\\test2.txt", "ab", pFisB) ;
// reatribui pFisB lui pFisT
// voi modifica acum fisierul test2.txt, cu ajutorul fluxului 'pFisT'

fprintf(pFisT, "\n Un text oarecare\n pentru proba. Fisier
binar") ;
if(feof(pFisT)) perror(sirEroareB) ; // test de EOF.
if(feof(pFisB)) perror(sirEroareB) ; // test de EOF.
}

// test eroare indicator fisier text: dupa operatii
if(!fgetpos(pFisT, &pozT))
    printf("\n\n Indicatorul de pozitie al fisierului text
(dupa): %u", pozT) ;
else {
    if(ferror(pFisT)) {
        perror(sirEroareT) ;
        exit(1) ; // iesire neconditioanata din program
    }
}

// test eroare indicator fisier binar: dupa operatii
if(!fgetpos(pFisB, &pozB))
    printf("\n Indicatorul de pozitie al fisierului binar
(dupa): %u", pozB) ;
else {
    if(ferror(pFisB)) {
        perror(sirEroareB) ;
        exit(1) ; // iesire neconditioanata din program
    }
}
} // end_else

// inchiderea fluxurilor
fclose(pFisT) ;
fclose(pFisB) ;

// sfarsit
printf("\n\n Dati un caracter alfa-numeric... ");
getch() ;
}

```

## 6.2. Probleme rezolvate

### 6.2.1. Enunțuri

1. Să se scrie un program ce compară două fișiere text, ale căror nume se dau de la tastatură și afișează prima linie diferită.
2. Să se scrie un program care citește un sir de numere de la tastatură și apoi scrie în fișierul "fisier3.dat", pe o linie, câte două numere consecutive, inserând între ele suma lor.
3. Să se scrie un program care scrie într-un fișier numit "fisier3.dat" un număr n de înregistrări și care permite, de asemenea, căutarea unei înregistrări indicate la poziția introdusă de utilizator.
4. Să se numere câte caractere de fiecare tip (litere mici, litere mari, cifre și caractere neafisabile) sunt într-un fișier text, având numele "f.txt".

### 6.2.2. Soluții propuse

#### Problema 1:

Să se scrie un program ce compară două fișiere text, ale căror nume se dau de la tastatură și afișează prima linie diferită.



```
#include <stdio.h>
#include <stdlib.h>
#include <io.h>
#include <fcntl.h>
int main()
{
    FILE *pf1, *pf2;
    char sir1[100], sir2[100];
    char nume1[10], nume2[10];
    printf("\n DATI NUMELE PRIMULUI FISIER:\n");
    scanf("%s", nume1);
```

```

if((pf1=fopen(nume1,"r")) == NULL)
{
    printf("\n EROARE LA DESCHIDERE %s\n",nume1);
    exit(1);
}
printf("\n DATI NUMELE CELUI DE AL DOILEA
FISIER:\n");
scanf("%s", nume2);
if((pf2=fopen(nume2,"r")) == NULL)
{
    printf("\n EROARE LA DESCHIDERE %s\n",nume2);
    exit(1);
}
while( !feof(pf1) && !feof(pf2) )
{
    fgets(sir1,99,pf1);
    fgets(sir2,99,pf2);
    if(strcmp(sir1,sir2)!=0)
    {
        printf("\nprima linie diferita a celor doua fisiere este:%s\n",
sir1);
        exit(1);
    }
}
}

```

### Problema 2:

Să se scrie un program care citește un sir de numere de la tastatură și apoi scrie în fișierul "fisier3.dat", pe o linie, câte două numere consecutive, inserând între ele suma lor.



```

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<io.h>
FILE *pf;
int i, n, x[30], y[30], nr;
double a, b;
int main()
{
    clrscr();
    printf("\ndati nr. de elemente : "); scanf("%d", &nr);

```

```

for(i=0; i<nr; i++)
{
    printf("x[%d] = ",i+1);
    scanf("%d", &x[i]);
}
pf=fopen("fisier3.dat","w");
for(;n<nr;n++)
{
    a = x[n] + x[n+1];
    fprintf(pf, "%d %4.2f %d\n", x[n], a, x[n+1]);
}
fclose(pf);
getch();
}

```

### Problema 3:

Să se scrie un program care scrie într-un fișier numit "fisier3.dat" un număr *n* de înregistrări și care permite, de asemenea, căutarea unei înregistrări indicate la poziția introdusă de utilizator.



```

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<io.h>

typedef struct inreg{
    int nr;
    char nume[20];
    char prenume[20];
    char tel[7];
}elev;

elev el;
FILE *pf;
unsigned int i, x, n;
long int ofs;

int main()
{
    clrscr();
    pf = fopen("fisier3.dat","w+");
    printf("\ncate inregistrari introduceti n = ");
    scanf("'%d", &n);
}

```

```

for(i=0;i<n;i++)
{
    printf("Numarul elevului : ");
    scanf("%d",&el.nr);
    printf("Numele elevului : ");
    scanf("%20s",&el.nume);
    printf("Prenumele elevului : ");
    scanf("%20s",&el.prenume);
    printf("Nr.de telefon : ");
    scanf("%7s",&el.tel);
    fflush(stdin);
    fwrite(&el,sizeof(elev),1,PF);
    puts("\n");
}
// citire pozitie inregistare
printf("\n ce inregistrare cautam ? ");
scanf("%u", &x);

// deplasare
ofs=(x-1)*sizeof(elev);
fseek(PF,ofs,0); // 0 = SEEK_SET - inceputul fisierului

fread(&el, sizeof(elev), 1, PF);
printf("\n\n intregistrarea este : \n %d %s %s %s", el.nr,
el.nume, el.prenume, el.tel);
fclose(PF);
getch();
}

```

#### Problema 4:

Să se numere câte caractere de fiecare tip (litere mici, litere mari, cifre și caractere neafisabile) sunt într-un fișiere text, având numele "f.txt".



```

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
int main()
{
    FILE *f;
    int p, b, d, g;
    char ch;
    clrscr();
    p=b=d=g=0;

```

```
if((f=fopen("f.txt","r"))==NULL)
{
    printf("\n nu se poate deschide fisierul F.TXT");
    exit(1);
}
do
{
    ch=fgetc(f);
    if(islower(ch)) p++;
    if(isupper(ch)) b++;
    if(isdigit(ch)) d++;
    if(!isprint(ch)) g++; // daca ch nu este afisabil
}while(ch!=eof(f));
fclose(f);
printf("\n numarul de litere mici : %d", p);
printf("\n numarul de litere mari : %d", b);
printf("\n numarul de cifre : %d", d);
printf("\n numarul de caractere neafisabile : %d", g);
getch();
}
```

### **6.3. Probleme propuse spre rezolvare**

**1.** Fiind dat un fișier să se determine numărul de linii, linia de lungime maximă și lungimea fișierului. În plus, fiind dat un cuvânt, să se determine numărul de apariții ale cuvântului în fișier. Numele fișierului și cuvântul se vor da de la tastatură.

**2.** Să se programeze o funcție C++ care să realizeze:

a) crearea unui fișier secvențial cu structura înregistrării:

NUME, PRENUME, MEDIA1, MEDIA2, MEDIA3

b) să afișeze apoi înregistrările sortate după media generală pe persoană.

**3.** Se citesc din fișierul text 'FISIER.TXT' un sir de numere întregi aflate toate pe prima linie a fișierului, separate între ele prin spații. Se cere să se determine câte din numerele citite sunt mai mici decât media aritmetică a tuturor numerelor. Rezultatul se va afișa pe ecran.

**4.** Scrieți un program C++ care citește din fișierul text 'NUMERE.TXT' mai multe numere naturale de cel mult trei cifre fiecare și afișează pe ecran câte numere sunt prime, iar răsturnatele lor sunt tot numere prime. Numerele sunt scrise pe prima linie a fișierului, cu câte un spațiu între ele.

**5.** Să se scrie într-un fișier text următoarea piramidă :

A

A A

A A A

.....

Numărul de caractere de pe ultima linie se va citi de la tastatură.

**6.** Scrieți un program care să numere cuvintele și propozițiile unui text. Textul este încheiat de caracterul "sfârșit de fișier". Cuvintele sunt separate prin

virgulă și spațiu, iar propozițiile se termină cu unul dintre caracterele punct, semnul exclamării sau semnul întrebării. Un cuvânt poate începe printr-o literă sau printr-o cifră iar în interiorul cuvintelor se acceptă doar: literele, cifrele și caracterul cratimă (-).

7. Se dau un text și un cuvânt ce urmează să fie căutat în acest text. Să se scrie un program care să numere aparițiile cuvântului căutat și în locul lui să introducă un alt cuvânt, de asemenea precizat. În final, se va afișa textul astfel modificat.
8. Dintr-un text considerat cunoscut să se contorizeze numărul de caractere, numărul de cuvinte și numărul de linii. Pentru fiecare cuvânt din text (identificat printr-un număr de ordine) să se afișeze lungimea (numărul de caractere din care este compus).
9. Să se realizeze descrierea în limbajul C a informațiilor referitoare la un grup de persoane (numărul maxim de persoane în grup este 1000). Despre fiecare persoană se folosesc următoarele informații: nume, prenume, adresă, sex, greutate, înălțime, vîrstă, culoare a părului și a ochilor. Să se scrie un program care stochează informațiile despre persoane pe disc magnetic, într-un fișier care să poată fi citit dacă se dorește afișarea acestora într-o formă convenabilă. Programul se poate extinde astfel încât să solicite introducerea de la tastatură a numelui unei persoane și să afișeze informațiile despre toate persoanele cu numele respectiv (dacă există asemenea persoane). Dacă de la tastatură se introduce textul "oricare", programul va afișa informațiile despre toate persoanele, indiferent de nume.
10. Să se scrie un program care să citească patru grupuri de caractere, notate a, b, c, d. Fiecare grup este situat pe câte o linie distinctă. Programul calculează următoarele expresii și le afișează sau stochează într-o formă convenabilă:  
$$(a - b) \cup (c - d)$$

(a - b) - (c - d)

11. Să se scrie un program care, primind numele a cel mult patru fișiere precedate de numărul acestora, să formeze fișierul obținut prin concatenarea tuturor informațiilor din acestea. Fișierele conțin numere de tip real.
12. Să se scrie un program care compară, linie cu linie, conținutul a două fișiere text, tipărind numărul de ordine al liniilor în care apar deosebiri.
13. Să se scrie un program care să ordoneze un fișier ce conține elemente de tip structură după o cheie formată din 8 caractere, transferând apoi înregistrările astfel ordonate în alt fișier.
14. Se consideră un fișier care conține informații referitoare la mărfurile prezentate în cadrul unui catalog promovațional. Pentru fiecare marfă se precizează (pentru simplitate) doar codul alfanumeric asociat și prețul exprimat în lei. Să se scrie o funcție care să furnizeze ca rezultat marfa cu cel mai mic preț din întregul catalog (codul alfanumeric asociat acesteia).
15. Se consideră un fișier în care se pot păstra ca informații numere întregi. Să se scrie o funcție care depune în acest fișier numerele Fibonacci ce nu depășesc o valoare dată ca parametru, notată **n**.
16. Se consideră un fișier în care sunt înscrise următoarele informații referitoare la o grupă de studenți: nume student, prenume student, media obținută de student în sesiunea de examene curentă. Să se scrie funcții care să realizeze următoarele operații:
  - să afișeze media și numele studentului cu media cea mai mare
  - să determine media studentului de la mijlocul fișierului și să semnalizeze situația în care fișierul conține un număr impar de articole

- să ordoneze crescător articolele fișierului după valoarea mediilor și să afișeze media și numele studentului aflat pe ultima poziție din fișier în urma acestei operații

17. Pentru fiecare produs realizat de o anumită firmă se păstrează evidența următoarelor informații: cod beneficiar, cod produs, cantitate livrată, valoare. Să se scrie un program care să realizeze următoarele operații:

- să creeze un fișier notat **f** care să conțină informații referitoare la toate produsele firmei
- să creeze un fișier notat **g** în care să stocheze toate articolele fișierului **f** a căror valoare este mai mare decât o valoare **x** precizată
- să creeze un fișier notat **h** în care să stocheze articolele fișierului **f** pentru care cantitatea livrată este mai mică decât o valoare **y** precizată

18. Articolele unui fișier conțin următoarele informații referitoare la lucrătorii din cadrul unei secții:

- număr marcă
- nume lucrător
- cod operație efectuată
- număr de execuții ale operației
- valoare manoperă pentru execuția unei operații

Dacă articolele sunt ordonate crescător după valoarea "număr marcă", să se creeze un fișier cu articole care conțin următoarele informații: număr marcă, nume lucrător, retribuția.

19. Se consideră un fișier cu informații despre locul nașterii unor persoane. Să se determine pentru fiecare localitate numărul de persoane născute în localitatea respectivă.

20. Să se scrie un program prietenos cu utilizatorul care să realizeze evidența operațiilor efectuate într-un magazin comercial pe baza următoarelor precizări:

- la fiecare vânzare se înregistrează lista mărfurilor cerute de cumpărător împreună cu cantitățile solicitate. Se presupune că există un fișier cu mărfurile disponibile și prețurile acestora. Programul va trebui să indice costul total al mărfurilor solicitate care există în magazin și lista mărfurilor ce nu sunt disponibile în cantitatea solicitată. Atunci când cumpărătorul achiziționează mărfurile existente se va face actualizarea stocului
- la fiecare aprovizionare se introduce o listă de mărfuri împreună cu cantitățile ce vor intra în magazie. Pentru mărfurile cu prețuri noi se va specifica și prețul pe bucătă
- la sfârșitul fiecărei săptămâni se va edita un raport ce va conține pentru fiecare produs codul și cantitatea disponibilă în stoc. De asemenea, se va afișa lista produselor cel mai des solicitate

**21.** Se dau fișierele:

f1 cu articole a1 = (a,b,c,d);

f2 cu articole a2 = (e,f,g,h,i);

f3 cu articole a3 = (j,k),

unde a,b,c,d,e,f,g,h,i,j, și k sunt numere reale. Să se creeze fișierele:

f4 cu articole a4 = (t1,t2,t3,t4)

și

f5 cu articole a5 = (u1,u2,u3,u4,u5),

unde:

t1 = max (a,b,c,d);

t2 = min (e,f,g,h,i);

t3 = max (j,k);

t4 = max (a,b,c,d,e,f,g,h,i,j,k);

u1= min (a,b,c,d);

u2= max (e,f,g,h,i);

u3= min (j,k);

u4= min (a,b,c,d,e,f,g,h,i,j,k);

u5= min (u1,u2,u3,u4).

**22.** Se consideră un fișier ce conține următoarele informații referitoare la produsele prezentate în cadrul unui catalog: codul firmei, codul produsului, denumirea produsului și cantitatea produsă. Se cere:

- să se ordeneze fișierul crescător după codul firmei, stabilindu-se apoi numărul de produse realizate de către fiecare firmă
- să se ordeneze fișierul crescător după codul produsului, stabilindu-se pentru fiecare produs numărul de firme care îl realizează

**23.** Scrieți un program care creează un fișier text prin combinarea informațiilor conținute în alte două fișiere de același tip. (În noul fișier apar alternativ: prima linie din primul fișier, prima linie din al doilea fișier, a doua linie din primul fișier, a doua linie din al doilea fișier etc.). Se va lua în considerare și cazul în care cele două fișiere conțin un număr diferit de linii.

**24.** Scrieți un program care să afișeze din fiecare linie a unui fișier text toate coloanele (caracterele conținute în pozițiile) **m** până la **n**, unde **m** și **n** se citesc de la tastatură.

**25.** Scrieți un program care să afișeze conținutul unui fișier text astfel:

- se afișează primele 20 de linii din fișier
- se așteaptă introducerea unui caracter de la tastatură
- dacă s-a apăsat tasta corespunzătoare literei **s** (cu semnificația de **stop**), programul se încheie (se abandonează afișarea)
- dacă se apasă orice altă tastă (corespunzătoare altor litere decât **s**), se vor afișa următoarele 20 de linii din fișier

**26.** Să se afișeze numărul de numere de pe fiecare linie a unui fișier text cu numele citit de la tastatură.

Exemplu: Dacă numele fișierului citit este F1.TXT iar conținutul acestui fișier este:

```
12 4 67 43 878  
2 7 8 5 3 2 89 123
```

valorile afişate vor fi 5, 8, 1.

**27.** În fișierul 'X.txt' se găsesc  $n$  perechi de numere întregi. Fișierul conține: pe primul rând valoarea lui  $n$ , apoi, pe fiecare din următoarele  $n$  rânduri elementele unei perechi separate printr-un spațiu. Să se afișeze numărul perechilor cu proprietatea că media aritmetică a celor două numere din pereche este egală cu o valoare dată  $M$  (unde  $M$  se citește de la tastatură).

**28.** Fișierul text 'Triunghi.txt' conține pe fiecare linie câte trei numere separate prin câte un spațiu. Pentru fiecare dintre aceste triplete să se verifice dacă pot reprezenta laturile unui triunghi (sunt pozitive și suma a oricare două este mai mare decât al treilea număr), iar în caz afirmativ să se calculeze aria triunghiului folosind formula lui Heron. Rezultatele se vor scrie în fișierul 'Arii.txt': Pentru fiecare triplet se va scrie în fișierul 'Arii.txt', pe un rând, aria triunghiului sau mesajul 'nu e triunghi'.

**29.** Fiind dat un fișier care conține mai multe linii de text de lungime variabilă, scrieți un program care afișează linia (liniile) de lungime maximă.

**30.** Se citește o matrice din fișierul text 'Matrice.in'. Fișierul conține :

- pe primul rând numărul  $m$  de linii și numărul  $n$  de coloane ale matricii
- pe fiecare din următoarele  $m$  rânduri, elementele unei linii a matricii, separate prin spații

Să se interschimbe între ele două linii date  $L_1$ ,  $L_2$ , scriindu-se matricea rezultată în fișierul 'Matrice.out' (elementele fiecărei linii a matricii pe rând).

**31.** Scrieți un program care generează toate numerete prime strict mai mici decât  $x$  ( $x$  număr natural). Valoarea variabilei  $x$  se citește de la tastatură. Numerele prime generate vor fi scrise în fișierul text 'BAC.txt', câte unul pe linie.

*Exemplu:* pentru  $x=10$  se vor scrie în fișier numerele 2, 3, 5 și 7.

**32.** Realizați un program care concatenează două fișiere text ("lipește" conținutul celui de-al doilea la sfârșitul primului).

**33.** Se dă un fișier care conține mai multe linii de text. Să se tipărească, pentru fiecare linie în parte, multimea caracterelor distincte din linia respectivă. Rezultatele se vor afișa pe ecran.

**34.** Scrieți un program care citește de la tastatură un număr n ( $n \leq 10^9$ ) și creează fișierul text 'BAC.TXT' care să conțină toate pătratele perechi mai mici sau egale cu n, despărțite între ele prin câte un spațiu. Algoritmul corespunzător programului trebuie să aibă în vedere o prelucrare cât mai eficientă ca timp de executare și spațiu de memorie utilizat.

*Exemplu:* pentru n=68, conținutul fișierului 'BAC.TXT' este următorul:

0 1 4 9 16 25 36 49 64

**35.** Scrieți un program care citește de la tastatură un număr natural n, ( $0 < n \leq 15$ ) și crează un fișier text cu numele 'A.TXT' ce conține pe prima linie un caracter A, pe a doua linie două caractere A nedespărțite prin spații, pe linia a treia linie trei caractere A nedespărțite prin spații etc. Ultima linie a fișierului trebuie fie linia a n-a care să conțină n caractere A nedespărțite prin spații.

*Exemplu:* dacă se citește n=4, fișierul 'A.TXT' creat trebuie să fie:

A  
AA  
AAA  
AAAA

**36.** Scrieți un program eficient care citește de la tastatură un cuvânt de cel mult 20 de caractere format numai din litere mari și memorează în fișierul text 'BAC.TXT' toate cuvintele distincte ce se pot forma prin eliminarea unui singur caracter din cuvântul dat. Fiecare cuvant va fi scris pe câte o linie a fișierului, fără spații între litere.

*Exemplu:* pentru cuvântul MARA, fișierul ‘BAC.TXT’ va conține următoarele cuvinte (nu neapărat în această ordine): MAA ARA MRA MAR

37. Fișierul ‘BAC2000.TXT’ este format din două linii. Prima linie este formată dintr-un sir de cel puțin una și cel mult 100 de litere mari nedespărțite prin spații, pe linia a doua aflându-se cel puțin unul și cel mult 100 de numere nedespărțite prin spații. Numerele de pe a doua linie sunt valori întregi din intervalul [-1000, 1000]. Conținutul fișierului respectă întocmai precizările anterioare, nefiind necesare validări.

Scrieți un program C++ care să decidă dacă în fișier există tot atâtea litere pe prima linie câte numere se găsesc pe a doua linie. Se va afișa pe ecran mesajul “*Da*”, dacă sunt tot atâtea litere pe prima linie câte numere sunt pe a doua linie, sau mesajul “*Nu*” în caz contrar.

38. Se citește un sir de numere întregi din fișierul text ‘Sir.in’. Toate elementele sirului sunt scrise în fișier pe un singur rând, separate prin spații. Nu se cunoaște numărul elementelor. Să se scrie în fișierul ‘Sir.out’ elementele distincte ale sirului citit, “grupate” pe două rânduri: pe primul rând se vor scrie elementele pare în ordine crescătoare, iar pe al doilea rând elementele impare în ordine crescătoare. *Nu se va folosi nici un algoritm de sortare.*

39. O firmă de telecomunicații are un număr de abonați telefonici ale căror numere de telefon sunt formate din șase cifre, prima fiind obligatoriu diferită de zero. Firma a organizat un concurs. Premiul va fi oferit abonatului al cărui număr de telefon conține doar cifre pare, iar dintre acestea este maxim conform ordinii numerelor naturale. Datele de intrare se citesc din fișierul ‘Abonati.txt’, care conține pe fiecare rând numele și numărul de telefon al unui abonat, separate prin exact un spațiu.

Să se găsească și să se afișeze pe ecran numele și numărul abonatului câștigător. Dacă nu există un astfel de abonat, se va tipări un mesaj.

**40.** Pe prima linie a fișierului text 'BAC2000.TXT' se găsește o succesiune de cel puțin două și cel mult 2000 de caractere, caractere ce pot fi doar litere mari și spații. Scrieți un program C++ care citește de la tastatură un număr natural k ( $0 < k < 1000$ ) și stabilește dacă există în fișier vreo literă care apare de exact k ori. Programul care afișează pe ecran mesajul "Da" în cazul în care există cel puțin o literă cu proprietatea menționată și mesajul "Nu", în caz contrar.

*Exemplu:* dacă fișierul 'BAC2000.TXT' are următorul conținut

#### EXAMEN DE BACALAUREAT LA INFORMATICA

iar de la tastatură se citește numărul 8, atunci, deoarece litera A apare de exact 8 ori, se va afișa pe ecran mesajul "Da".

**41.** Fișierul 'Text.in' conține un text, pe fiecare rând fiind scrisă o frază. Cuvintele unei fraze pot fi separate între ele prin caracterele "spațiu", "virgulă" și "punct și virgulă". Să se tipărească în fișierul 'Cuvinte.out', cuvintele distincte din text și frecvențele lor de apariție în cadrul textului (pe fiecare rând se va scrie un cuvânt împreună cu frecvența sa, separate printr-un spațiu).

*Exemplu:*

Text.in                    e bine bine, e foarte bine

                          e extraordinar de bine

Cuvinte.out            e 3

                          bine 4

                          foarte 1

                          extraordinar 1

                          de 1

**42.** Se dă fișierul text 'Fr.in' care conține pe fiecare linie câte două numere naturale mai mici decât 35000, reprezentând numărătorul respectiv numitorul unei fracții simple.

a) Să se creeze fișierul 'Fr.out' care va conține pe fiecare rând numărătorii și numitorii fracțiilor ireductibile rezultate prin simplificarea fracțiilor din fișierul 'Fr.in'.

b) Pe ultima linie a fișierului ‘Fr.out’ să se scrie numărătorul și numitorul fracției rezultate prin adunarea fracțiilor din fișierul ‘Fr.in’. Fracția sumă se va da în formă ireductibilă.

*Exemplu :*

Pentru ‘Fr.in’ 2 6

12 18

50 20

fișierul ‘Fr.out’ va conține : 1 3

2 3

5 2

7 2

## 6.4. Teste grilă



6.4.1. Care dintre următoarele operații nu sunt posibile într-un fișier text ?

- a) Modificarea unor valori în fișier, fără folosirea altor fișiere sau structuri de date
- b) Testarea sfârșitului de rând în fișier
- c) Testarea sfârșitului de fișier
- d) Deschiderea pentru citire
- e) Deschiderea pentru adăugare de noi date la începutul fișierului



6.4.2. Fie un fișier identificat prin descriptorul `f` și deschis cu atributul "w". Fie, de asemenea, două variabile întregi `x` și `y`, ale căror valori sunt cunoscute. Care dintre instrucțiunile de mai jos pot fi executate astfel încât valorile celor două variabile să fie scrise în fișier fiecare pe alt rând?

- a) `fprintf(f,"%d\n%d", x, y);`
- b) `fprintf("%d\n%d", x, y, f);`
- c) `fprintf(f,"\\n%d%d\\n", x, y);`
- d) `fprintf ("\\n%d%d\\n", x, y, f);`
- e) `fprintf("\\n%d%d\\n", x, y, *f);`



6.4.3. Fie fișierul cu descriptorul `f` cu următorul conținut:

50 2
1.4 3
8

Ce se va afișa pe ecran în urma execuției programului următor?

```

#include <stdio.h>
int main()
{
    FILE *f;
    int a, b;
    float x, y;
    f=fopen("intrare.txt","r");
    fscanf(f,"%d %f\n", &a, &x);
    fscanf(f,"%f\n", &y);
    fscanf(f,"%d", &b);
    printf("\na=%d x=%f y=%f b=%d", a, x, y, b);
    fclose(f);
}

```

- a) a=50 x=2.000000 y=1.400000 b=8
- b) a=50 x=2.000000 y=1.400000 b=3
- c) a=50.000000 x=2.000000 y=1.4 b=3.000000
- d) a=50.000000 x=2.000000 y=1.4 b=8.000000
- e) Citirile din fișier vor eşua



#### 6.4.4. Fie programul următor:

```

#include <stdio.h>
int main()
{
    FILE *f, *g;
    int a, x, S;
    f = fopen ("in.txt", "r");
    g = fopen ("out.txt", "w");
    scanf ("%d", &a);
    while (!feof(f))
    {
        S=0;
        while ( S < a && !feof(f))
        {
            fscanf (f,"%d", &x);
            S+=x;
        }
        fprintf (g,"%d ",S);
    }
    fclose (f);
    fclose (g);
    printf ("\nS=%d", S);
}

```

Dacă de la tastatură se introduce valoarea 10, iar conținutul fișierului 'in.txt' este cel de mai jos, câte numere va scrie programul în fișierul 'out . txt'?  
?

4 6 3 2 6 15 1



6.4.5. Câte numere se vor găsi în fișierul "nr.txt" după execuția programului următor?

```
#include <stdio.h>
int main ()
{
    int v[9]={0,1,0,0,2,3,0,4,5},i;
    FILE *f;
    f=fopen("nr.txt","w");
    i=0;
    while (i<9)
    {
        while (v[i])
        {
            fprintf (f,"%3d",v[i]);
            i++;
        }
        fprintf (f,"%3d", 99) ;
        i++;
    }
    fclose (f) ;
}
```

- a) 4      b) 5      c) 8      d) 9      e)10



6.4.6. Presupunem că pe hard-disk există fișierul "car.txt" cu următorul conținut:

AbCdEfGhI

Știind că în conformitate cu standardul ASCII literele mari au coduri succesive începând cu 65 ('A' ← 65, 'B' ← 66, etc.), iar literele mici au coduri succesive începând cu 97 ('a' ← 97, 'b' ← 98, etc.), precizați care va fi conținutul fișierului "car2. txt" după execuția programului următor:

- a) bdfhj
- b) BDFHJ
- c) acegi
- d) ACEGI
- e) abcdefghij

```
#include <stdio.h>
int main ()
{
    char c1,c2;
    FILE *f,*g;
    f=fopen("car.txt","r");
    g=fopen("car2.txt","w");
    c1=96;
    while (!feof(f))
    {
        c2=fgetc(f);
        printf("%c",c2);
        if (c2>c1) fputc(c2-32,g);
        c1=c2;
    }
    fclose(f);
    fclose(g);
}
```



**6.4.7.** Deducreți ce valoare va afișa programul următor, presupunând că în fișierul cu descriptorul f se găsesc pe un rând, separate prin spații, numerele:

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

- a) 1
- b) 72
- c) -12
- d) Programul conține erori de sintaxă
- e) Nu se pot citi corect numerele din fișier

```
#include <stdio.h>
#include <math.h>
int main ()
{
    FILE *f;
    int s=1, i=0, a[20];
    f=fopen ( "numere . txt" , "r" );
    while ( ! feof ( f ) )
    {
        i++;
        fscanf(f,"%d ", &a[i]);
        if (a[i]) s*=abs(a[i]);
    }
    printf ("\n%d", s);
    fclose (f );
}
```



6.4.8. Presupunând că toate liniile fișierului cu descriptorul g conțin cel mult 100 de caractere, care este acțiunea programului următor?

```
#include <stdio.h>
int main()
{
    FILE *f, *g;
    char s [101];
    f=fopen("1.txt","a");
    g=fopen("2. txt", "r");
    while (!feof(g))
    {
        fgets(s,100,g);
        fputs (s, f);
    }
    fclose (f);
    fclose (g);
}
```

- a) Înlocuiește conținutul fișierului g cu conținutul fișierului f
- b) Înlocuiește conținutul fișierului f cu conținutul fișierului g
- c) Concatenează fișierul g la sfârșitul fișierului f
- d) Concatenează fișierul f la sfârșitul fișierului g
- e) Nici unul dintre cazurile anterioare



6.4.9. Deduceți ce valoare va afișa programul de mai jos, dacă fișierul text are următorul conținut:

3 3
1 2 3
4 5 6
7 8 9

```

#include <stdio.h>
int main ()
{
    FILE *f;
    int i, j, m, n, S=0, a[20][20];
    f=fopen ("conținut . txt", "r");
    fscanf(f,"%d %d", &m, &n);
    for (i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
        {
            fscanf (f,"%d ",&a[i][j]);
            if ((i+j)%2) S+=a[i][j];
        }
    }
    fclose (f);
    printf ("%d",S);
}

```

- a) 0
- b) 8
- c) 20
- d) 25
- e) programul este eronat



6.4.10. Se dă fișierul identificat prin descriptorul f, cu următorul

conținut:

**33 1 -45 18 6**

Ce instrucțiune trebuie scrisă în loc de "....." astfel încât programul următor să tipărească 85?

```

#include <stdio.h>
int main ()
{
    FILE *f;
    int x, y;
    f=fopen("valori.txt", "r");
    fseek(f,-6,2);
    fscanf(f,"%d", &y);
    .....
    fscanf(f,"%d", &x);
    printf("\n%d%d", x, y);
    fclose(f);
}

```

- a) `fseek(f,11,0);`
- b) `fseek(f,-2,2);`
- c) `f seek (f, 3,1);`
- d) `fseek(f,2,1);`
- e) `fseek(f,-3,2);`



6.4.11. Precizați ce nume se va găsi pe al cincilea rând în fișierul "propus.txt", după execuția programului de mai jos?

```
#include <stdio.h>
#include <string.h>
int main()
{
    FILE *f;
    int i = 0, j, k;
    char *aux;
    char *a[9]={"Marius", "Claudiu", "3rei-Sud-Est",
    "Daniel", "Vasile", "Dan", "Sinacdu", "2Pac"} ;
    while (a[i]) i++;
    for (k=j +1; k<i; k++)
        if ( strcmp(a[j],a[k]) > 0 )
    {
        aux=a[j]; a[j]=a[k]; a[k]=aux;
    }
    k = 0;
    f = fopen("propus . txt","w");
    while (a[k])
        fprintf (f,"%s\n", a[k++]);
    fclose(f);
}
```

- a) 2Pac
- b) Claudiu
- c) Dan
- d) Daniel
- e) Marius



6.4.12. Precizați care va fi conținutul fișierului "b. txt" după execuția programului, știind că fișierul "a. txt" are următorul conținut:

<b>11 2 13 4 15 6 17 8 19</b>
-------------------------------

```
#include <stdio.h>
#include <math.h>
int main()
{
    FILE *f,*g;
    int v[10];
    f=fopen("a.txt", "r");
    g=fopen("b.txt","w");
    fread(v,8,1,f);
    fwrite(v,6,1,g);
    fclose(f);
    fclose(g);
}
```

- a) 11 2 13 4 15 6
- b) 1 2 1 4 1 6
- c) 11 2 13
- d) 11 2 1
- e) Un alt conținut decât cele prezentate anterior

## 6.5. Răspunsuri la întrebările grilă

- 6.5.1. b)
- 6.5.2. a)
- 6.5.3. b)
- 6.5.4. e)
- 6.5.5. e)
- 6.5.6. b)
- 6.5.7. b)
- 6.5.8. c)
- 6.5.9. c)
- 6.5.10. a), b) și e)
- 6.5.11. d)
- 6.5.12. d)

## Sistemul de intrare/ieșire



**n** limbațul C++ există două sisteme de intrări/ieșiri:

- ❑ Unul tradițional, moștenit din limbațul C, în care operațiile de intrare/ieșire se realizează cu ajutorul unor funcții din biblioteca standard a sistemului;
- ❑ Unul propriu limbațului C++, orientat pe obiecte.

Sistemul de I/O orientat pe obiecte al limbațului C++ tratează în aceeași manieră operațiile de I/O care folosesc consola și operațiile care utilizează fișiere (perspective diferite asupra aceluiași mecanism).

La baza sistemului de I/E din C++ se află:

- *două ierarhii de clase* care permit realizarea operațiilor de I/O;
- *conceptul de stream* (stream-ul este un concept abstract care înglobează orice *flux* de date de la o sursă (canal de intrare) la o destinație (canal de ieșire), la un consumator. Sursa poate fi tastatura (intrarea standard), un fișier de pe disc sau o zonă de memorie. Destinația poate fi ecranul (ieșirea standard), un fișier de pe disc sau o zonă de memorie (figura 7.1).

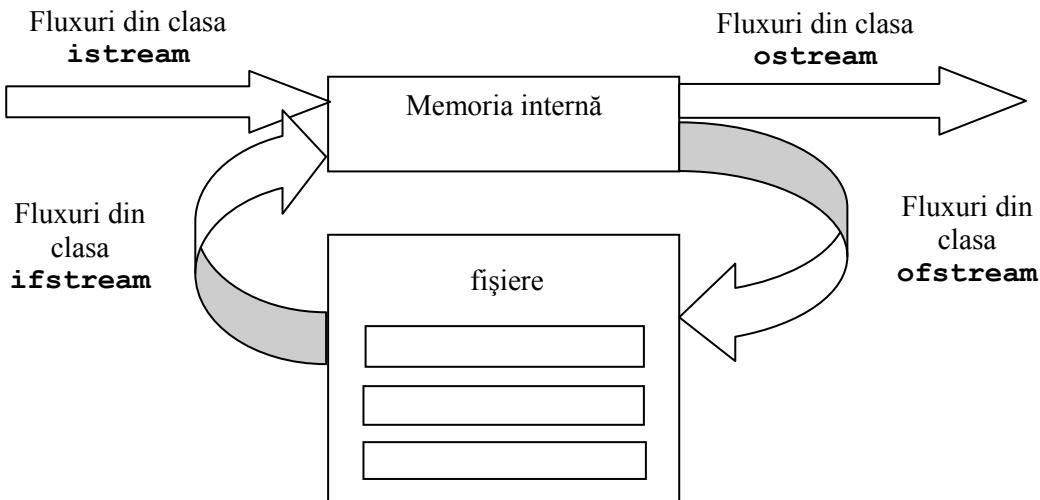


Figura 7.1. Stream-ul - baza sistemului de I/O

Avantajele utilizării **stream**-urilor sunt:

- Flexibilitate mare în realizarea operațiilor de I/O (mai mare decât în cazul folosirii funcțiilor din C);
- Posibilitatea de eliminare a erorilor care apar în mod frecvent la apelul funcțiilor de I/O obișnuite, când numărul specificatorilor de format diferă de cel al parametrilor efectivi;
- Posibilitatea de a realiza operații de I/O nu numai cu date de tipuri predefinite, ci și cu obiecte de tip abstract.

Biblioteca de clase de I/O a limbajului C++ utilizează moștenirea, polimorfismul și clasele abstrakte. Ierarhia are două clase de bază (figura 7.2.):

- Clasa virtuală **ios** care oferă informații despre fluxurile de date (variabile de stare, metode) și facilități tratarea erorilor;
- Clasa **streambuf** (clăsă prietenă cu **ios**), destinată operațiilor de I/O cu format.

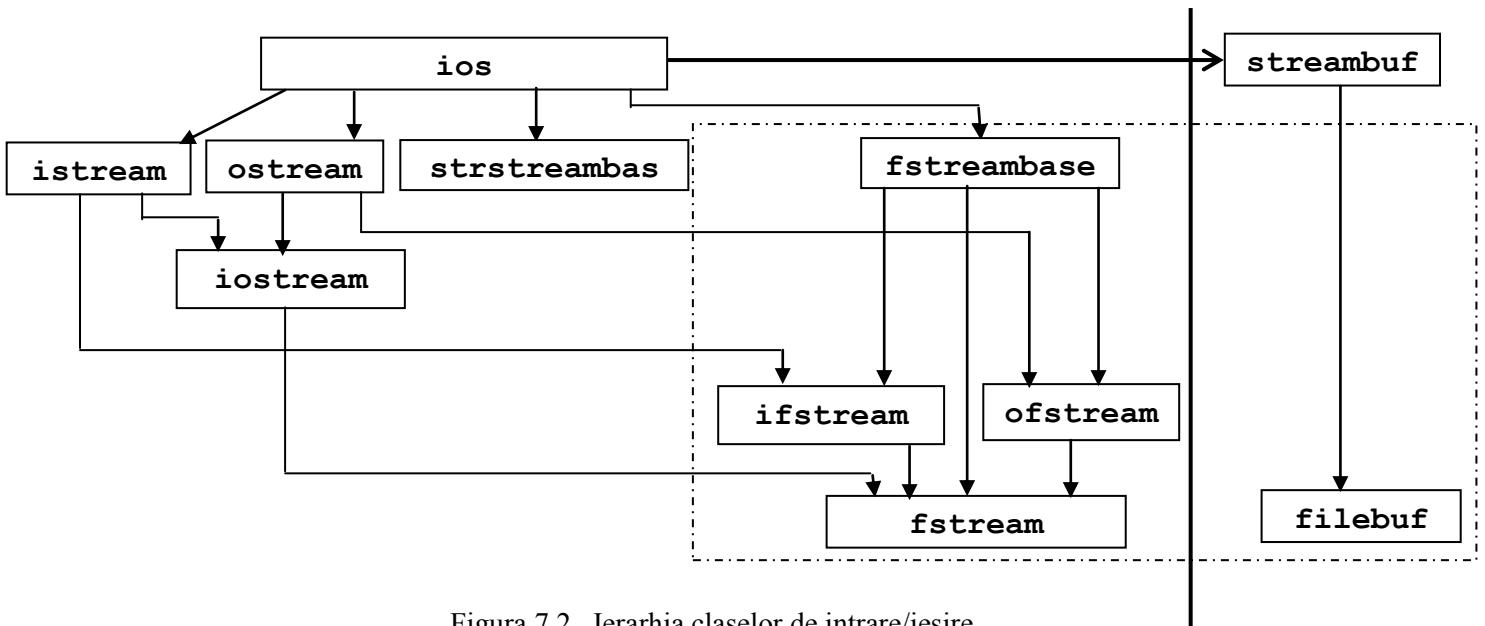


Figura 7.2. Ierarhia claselor de intrare/iesire

Utilizarea ierarhiilor de clase de mai sus implică includerea headerului *iostream.h*.

- Clasa **ios** are un pointer către **streambuf**. Are date membru pentru a gestiona interfața cu **streambuf** și pentru tratarea erorilor. Clasele derivate din clasa de bază **ios**:
- Clasa **istream**, care gestionează intrările: **class istream:virtual public ios**
- Clasa **ostream** gestionează ieșirile: **class ostream: virtual public ios**
- Clasa **iostream**, derivată din **istream** și **ostream**, gestionează intrările și ieșirile. Fiecărui flux de date î se asociază în memorie o zonă tampon numită **buffer**. Clasa furnizează funcții generale pentru lucrul cu zonele tampon și permite tratarea operațiilor de I/O fără a avea în vedere formatări complexe.

**class iostream:public istream, public ostream**

Claele **istream**, **ostream** și **iostream** sunt, fiecare, clase de bază pentru clasele derivate:



**class istream\_withassign:public istream**  
**class ostream\_withassign:public ostream**  
**class iostream\_withassign:public iostream**

Clasele cu sufixul `_withassig`n furnizează următoarele *fluxurile predefinite* (instanțe), deja cunoscute:

1. `cout` (console output) obiect al clasei `ostream_withassig`n, similar fișierului standard de ieșire definit de pointerul `stdout` (în C). Se folosește cu operatorul `insertor`, supraîncărcat pentru tipurile predefinite:

**Exemplu:** `int n; cout<<n;`

Convertește din format binar în zecimal valoarea lui `n` și trimită (inserează) în fluxul `cout` caracterele corespunzătoare fiecărei cifre obținute.

2. `cin` (console input) obiect al clasei `istream_withassig`n, similar fișierului standard de intrare definit de pointerul `stdin` (în C). A fost folosit cu operatorul `extractor`, supraîncărcat pentru tipurile predefinite:

**Exemplu:** `int n; cin>>n;`

Extrage din fluxul de intrare caracterele corespunzătoare, le convertește din zecimal în binar și le depune în memorie.

3. `cerr`: flux de ieșire conectat la ieșirea standard pentru erori (`stderr` în C) (fără buffer intermediu).
4. `clog`: flux de ieșire conectat la ieșirea standard pentru erori (fără buffer intermediu).

➤ Clasa `streambuf` este clasă prietenă cu `ios`. Ea furnizează funcții generale pentru lucrul cu zonele tampon (buffere) și permite tratarea operațiilor de I/O fără formatori complexe. Din clasa `streambuf` deriva clasa `filebuf`.

Să urmărim care este rolul buffere-lor (zonă tampon asociată fiecărui flux de date), prin următorul exemplu:



**Exemplu**  
**int main()**  
{

```
int a; double x; char sir[20];
cin>>a>>x>>sir;
cout<<a<<' '<<x<<endl;
cout<<sir<<endl;
```

}

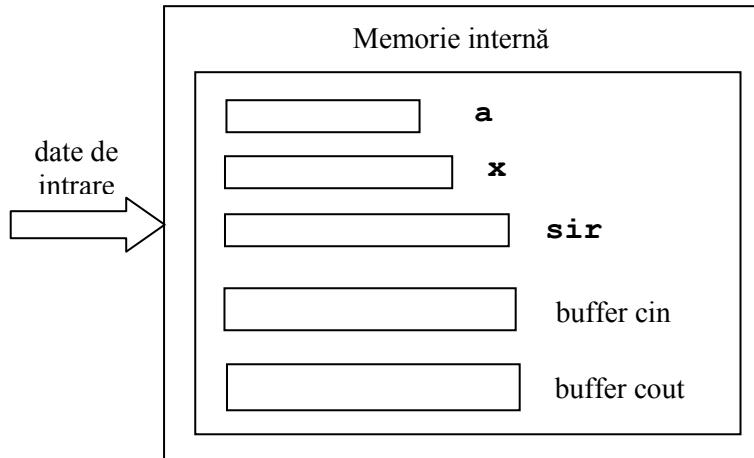


Figura 7.3. Rolul buffere-lor în operațiile de I/O

Să urmărim cum se realizează transferul informației în cazul operațiilor multiple:

`cin>>a>>x>>șir;`

`cin>>a;`

Compilatorul verifică dacă numărul introdus este întreg (se oprește când întâlnește altceva decât o cifră: în acest caz - un blanc). Îl citește printr-o metodă a tipului int, îl convertește în binar și îl transmite în memoria internă în spațiul rezervat pentru variabila a. În încheiere, `cin>>a` devine `cin`.

`cin>>a>>x;` // fluxul de date se transmite și lui x

Presupunem că am introdus de la tastatură 325 -17.45e5 exemplu de sir, valorile pentru a, x și sir. La apăsarea tastei Enter, se transmite către sistemul de operare un octet cu semnificația de sfârșit introducere date. Sistemul de operare transmite un semnal zonei tampon și abia acum se transmite valoarea (fluxul de date) lui a. La citirea sirului de caractere (până la primul blank), se transferă octet cu octet (i se adaugă automat caracterul NULL), din zona tampon în memoria internă.

#### Observații:

1. Stream-ul este secvențial.
2. El poate realiza intrări/ieșiri formatare.

3. Este bine că în cazul unui flux de intrare să se verifice mai întâi dacă în zona tampon se găsește ceva. Dacă nu, se poziționează cursorul la începutul zonei tampon.
4. Informația de la buffer către memorie este gestionată prin program; informația de la zona tampon către alte periferice este gestionată de către sistemul de operare.

## 7.1. Testarea și modificarea stării unui flux

Clasa `ios`, clasă de bază a ierarhiei claselor de I/O, *definește o mulțime de tipuri, variabile și metode comune tuturor tipurilor de stream-uri.*

Starea unui stream (rezultatul ultimului acces la acesta) este păstrată în **cuvântul de stare**, care este **dată membră** a clasei `ios`. Fiecarei instanțieri a unei clase de intrare/ieșire i se asociază propriul cuvânt de stare (o mulțime de indicatori de stare), care păstrează toate informațiile aferente erorilor apărute în cursul operațiilor cu stream-ul. Acești indicatori sunt memorati la nivel de bit în data membru `state`.



```
class ios
//.....
protected:
    int state; // păstrează la nivel de bit valorile indicatorilor de stare
public:
    enum io_state{
        goodbit=0x00, // ultima operație de intrare/ieșire corectă
        eofbit=0x01, // s-a întâlnit sfârșitul de fișier într-o operație de
                      // intrare (lipsa caracterelor disponibile pentru citire)
        failbit=0x02, // setat dacă ultima operație de intrare/ieșire a
                      // eșuat; stream-ul respectiv nu va mai putea fi folosit în operații de
                      // I/O până când bitul nu va fi șters
        badbit=0x04, // ultima operație de intrare/ieșire invalidă; în
                      // urma resetării flag-ului este posibil ca stream-ul să mai poată fi
                      // utilizat
        hardbit=0x08 // eroare irecuperabilă
    };
};
```

**Testarea valorii cuvântului de stare** asociat unui stream se realizează cu ajutorul **metodelor**:

- `int good();` // Returnează valoare diferită de zero dacă cuvântul de stare este 0
- `int eof();` // Returnează valoare diferită de zero dacă eofbit este setat
- `int fail();` // Returnează valoare diferită de zero dacă failbit, hardbit sau badbit sunt setați
- `int bad();` // Funcționează ca metoda fail, dar nu ia în considerare flagul failbit.

**Modificarea valorii cuvântului de stare** asociat unui stream se realizează cu ajutorul **metodelor**:

- `void clear(int i=0);`

Setează data membru state la valoarea parametrului (implicit 0). Este capabilă să șteargă orice flag, cu excepția flag-ului hardfail.

**Exemplu:** `a.clear(ios::badbit);`

Setează bitul failbit și anulează valorile celorlați biți din cuvântul de stare a fluxului a. Pentru ca ceilalți biți să rămână nemodificați, se apelează metoda rdstate.

**Exemplu:** `a.clear(a.rdstate()|val_f);`

Se setează un singur flag, lăsându-le nemodificate pe celelalte; val\_f are una din valorile definite de enumerarea io\_state.

- `int rdstate();`

Returnează valoarea cuvântului de stare, sub forma unui întreg (valoarea datei membru state).

**Exemplu:** `a.clear(ios::badbit | a.rdstate());`

Setează bitul failbit, fără a modifica valorile celorlați biți:

**Testarea cuvântului de stare** se poate realiza și prin folosirea **operatorilor ! și void\*** :

- Operatorul ! este supraîncărcat printr-o funcție membră, cu prototipul:

`int operator !();`

care returnează 0 dacă un bit de eroare din cuvântul de stare este setat (ca și metoda fail).

**Exemplu:**

```
if (!cin)          //echivalent cu if (!cin.good())
    cout<<"Bit de eroare setat în cuvântul de stare!\n";
else    cin>>a;
```

- Operatorul **void\*** convertește stream-ul într-un pointer generic. Conversia are ca rezultat zero dacă cel puțin un bit de eroare este setat:

```
operator void *();
```

**Exemplu:** O construcție de forma: `cin>>s;` are ca valoare o referință la stream-ul `cin`, din clasa `istream`. Această referință poate fi utilizată sub forma: `if (cin>>s)...`

Pentru scoaterea unui stream din starea de eroare, fie dispare cauza erorii, fie trebuie șterse flagurile care semnalizează eroarea.

## 7.2. Formatarea datelor din fluxurile de intrare/ieșire

Unul dintre principalele avantaje oferite de sistemul de I/O din limbajul C++ îl reprezintă ignorarea aspectului formatării, folosindu-se o **formatare implicită**. În plus, se permite definirea unei formatări specifice pentru o anumită aplicație. Așa cum s-a subliniat în cazul cuvântului de eroare, cuvântul de format poate fi privit ca un întreg, pentru care fiecare bit reprezintă o constantă predefinită din clasa `ios`. În cadrul acestui cuvânt sunt

definite câmpuri de biți (cuvântul de format este dată membră care conține un număr de indici ce sunt biți individuali).



```
class ios {  
//.....  
protected:  
    long flag_x; // păstrează la nivel de bit indicatorii de format  
    int x_width; // numărul de caractere utilizate pentru afișarea  
    unei valori pe ecran  
public:  
enum  
{skipws = 0x0001,           // salt peste spațiile albe de la intrare  
left = 0x0002,              // aliniere la stânga la ieșire  
right = 0x0004,             // aliniere la dreapta la ieșire  
internal = 0x0008,          // aliniere după semn sau specifikator al  
    bazei la ieșire  
dec = 0x0010,                // conversie în baza 10  
oct = 0x0020,                // conversie octală la intrare/ieșire  
hex = 0x0040,                // conversie hexa la intrare/ieșire  
showbase = 0x0080,           // afișarea bazei la ieșire  
showpoint = 0x0100,          // afișarea punctului zecimal pentru numere reale  
    la ieșire  
uppercase = 0x0200,           // afișarea cu majuscule a cifrelor hexa și a literei  
    E la ieșire.  
showpos = 0x0400,             // afișarea semnului + pentru numerele pozitive la  
    ieșire  
scientific = 0x0800,           // folosirea formatului exponențial (științific) pentru  
    numerele reale  
fixed = 0x1000,               // folosirea formatului în virgulă fixă pentru numere  
    reale  
unitbuf = 0x2000,             // golește zona tampon după fiecare ieșire  
stdio=0x4000                 // golește "stdout" și "stdin" după fiecare inserare  
};  
};
```

În figura 7.4. sunt prezentate numele câmpurilor de biți (acolo unde este cazul). În cadrul fiecărui câmp de biți (**adjustfield**, **basefield**, **floatfield**) un singur bit poate fi activ.

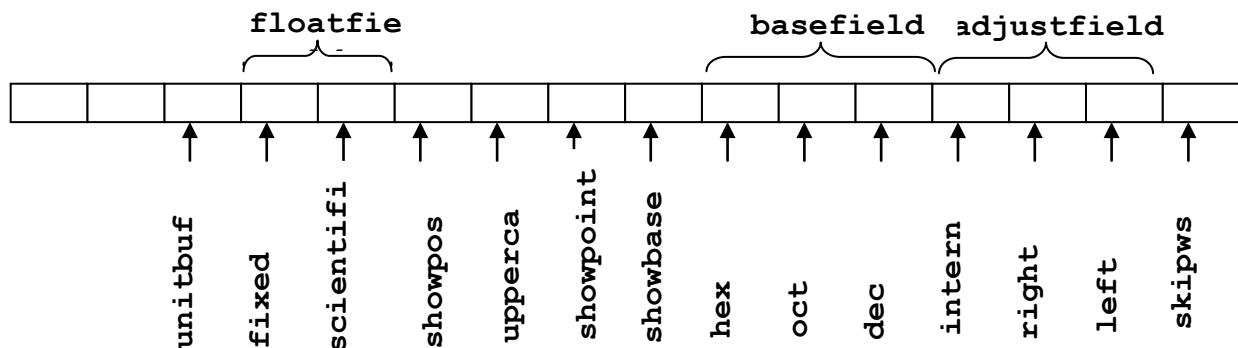


Figura 7.4. Câmpurile de biți din cuvântul de stare

**Modificarea cuvântului de format se poate realiza în următoarele moduri:**

1. Cu ajutorul *manipulatorilor* (cu sau fără parametri);
  2. Cu ajutorul unor *funcții membre* ale claselor *istream* sau *ostream*.

### 7.2.1. Formatarea prin manipulatori

Manipulatorii sunt *funcții speciale*, asemănătoare operatorilor, care pot fi folosite împreună cu operatorii de inserție într-un flux de ieșire sau de extractie dintr-un flux de intrare, în scopul modificării caracteristicilor formatului informațiilor de intrare/ieșire. Manipulatorii furnizează, ca rezultat, fluxul obținut în urma acțiunii manipulatorilor, ceea ce permite ca aceștia să fie tratați ca informații de transmis. Manipulatorii permit, deasemenea, înlătuirea operatorilor insertori sau extractorii care utilizează formate diferite.

Manipulatorii pot fi:

- Manipulatori fără parametri
  - Manipulatori cu parametri

### 7.2.1.1. Manipulatori fără parametri

Prototipul manipulatorilor fără parametri este:



```
ostream & nume_manipulator(ostream &);  
istream & nume_manipulator(istream &);
```

Manipulatorii fără parametri (prezentați în tabelul 7.1.) se folosesc astfel:



```
flux_ieșire<<manipulator;  
flux_intrare>>manipulator;
```

Tabelul 7.1

Manipulator	Intrare/ Ieșire	Acțiune
dec	I/O	Formatează datele numerice în zecimal (activează bitul de conversie zecimală)
hex	I/O	Formatează datele numerice în hexa (activează bitul de conversie hexazecimală)
oct	I/O	Formatează datele numerice în octal (activează bitul de conversie octală)
ws	I	Ignoră caracterele "spații albe" (activează bitul de salt peste spațiile albe)
endl	O	Afișează (inserează) un caracter '\n' și eliberează fluxul
ends	O	Inserează un caracter null, de sfârșit de flux (\0)
flush	O	Videază (golește) buffer-ul, eliberează fluxul

### 7.2.1.2. Manipulatori cu parametri

Prototipul manipulatorilor fără parametri (prezentați în tabelul 7.2.) este:



**istream & manipulator (argument);  
ostream & manipulator (argument);**

Tabelul 7.2.

Manipulator	Intrare/ Ieșire	Acțiune
<b>setbase(int baza)</b>	I/O	Stabilește baza de conversie
<b>resetiosflags(long f)</b>	I/O	Atribuie valoarea 0 tuturor biților indicați de argument, lăsând restul biților nemodificați (dezactivează indicatorii specificați de f)
<b>setiosflags (long f)</b>	I/O	Atribuie valoarea 1 tuturor biților indicați de argument, lăsând restul biților nemodificați (activează indicatorii specificați de f)
<b>setfill (int c)</b>	I/O	Definește caracterul de umplere (cel implicit este spațiul liber, blank-ul)
<b>setprecision (int p)</b>	I/O	Definește precizia pentru numerele reale
<b>setw (int w)</b>	I/O	Definește lățimea câmpului (numărul de octeți care vor fi citiți sau afișați)

Utilizarea manipulatorilor impune includerea header-ului **iomanip.h**.

**Exercițiu:** Exemplificarea modului de folosire a manipulatorilor pentru intrări/ieșiri formatare.



```
#include <iostream.h>
#include <iomanip.h>
int main()
{
    int a,b,c;double alfa,x,y;long ll;char xx,yy;
    float u,v; unsigned int w; unsigned char ab,db;
    a=125,b=10;ll=100*a*b;
    cout<<"100*a*b="<<ll<<endl; //100*a*b=-6072
    alfa=75.50;y=1.345678;
    xx=35;yy=6; x=y/alfa-xx/yy;
    cout<<"x="<<x<<endl; //x=-4.982176
    //setarea mărimea câmpului de afişare
    cout<<"x="<<setw(8)<<x<<endl; //x=-4.982176
    cout<<"x="<<setw(20)<<x<<endl; //x= -4.982176
    //setarea lungimii câmpului de afişare și a caracterului de umplere
    cout<<"x="<<setw(20)<<setfill('*')<<x<<endl;
    //x=*****-4.982176
    //precizie
    cout<<"x="<<setw(8)<<setprecision(2)<<x<<endl;
    //x=***-4.98
    cout<<"x="<<setw(8)<<setprecision(0)<<x<<endl;
    //x=-4.982176
    //afisarea octetului semnificativ var & 0377 si a celui mai
    putin semnificativ (var>>8) & 0377
    w=34; ab='34' & 0377; db=(34>>8)&0377;
    cout<<"w="<<w<<" ab="<<ab<<" db="<<db<<endl;
    u=2579.75; v=12345.567E+10;
    //formatare in octal si hexa
    cout<<"a in baza 10="<<a<<" in octal="<<oct<<a<<"  

    in hexa="<<hex<<a<<endl;
    //a in baza 10=125 in octal=175 in hexa=7d
    cout<<"b="<<dec<<b<<endl; //b=10
}
```

### 7.2.2. Formatarea prin metode

Formatarea fluxurilor de intrare/ieșire se poate realiza și prin **metode** ale clasei **ios**.

- Funcția membră **setf** permite modificarea cuvântului de stare a formatării. Ea este supradefinită astfel:

**long setf (long);**

Primește ca parametru un număr întreg long și setează pe 1 biți specificați prin argument. Returnează valoarea anterioară a cuvântului de stare a formatării. Ea poate fi folosită numai pentru flag-urile care nu fac parte dintr-un câmp de biți.

**Exemplu:** cout.setf(ios:: showpos); // setează bitul showpos pe 1

**long setf (long flags, long field);**

Setează pe 1 biți specificați prin primul argument, doar în câmpul de biți definit ca al doilea argument. Ea modifică pe 0 toți biții din câmpul respectiv, după care setează (pe 1) bitul respectiv.

**Exemplu:**

```
cout.setf(ios:: showpos | ios:: uppercase | ios:: internal) // toți biții vor fi setați pe valoarea 1
```

```
cout.setf(ios:: hex, ios:: basefield) // setează indicele hex din câmpul basefield
```

```
cout.setf(ios:: skipws | ios:: hex | ios:: showbase, ios:: basefield)
```

- Funcția membră **fill** permite testarea sau modificarea caracterului de umplere și returnează codul caracterului curent de umplere:

**int fill();**

**Exemplu:** cout.fill(); // întoarce codul caracterului de umplere pt. cout

**int fill(char);**

Setează noul caracter de umplere și returnează codul vechiului caracter de umplere.

**Exemplu:** cout.fill('#'); // setează un alt caracter de umplere ('#').

- Funcția membră **precision** permite testarea sau modificarea preciziei. Numim precizie numărul de cifre semnificative (în GNU) sau numărul de cifre după virgulă (în TurboC)

```
int precision ( );
```

Returnează valoarea actuală a preciziei numerice.

```
int precision (int);
```

Setează precizia la valoarea parametrului și returnează vechea valoare a preciziei.

- Funcția membră **width** returnează valoarea actuală a lungimii câmpului de date sau setează valoarea lungimii câmpului de date.

```
int width ( );
```

```
.int width (int);
```

**Exemplu:** cout.width (10); //data se va afișa într-un câmp de cel puțin 10 caractere.

Astfel, dacă argumentul este 0, la inserția într-un flux de ieșire se transmit în stream atâtia octeți căți are data respectivă. Dacă argumentul este diferit de 0, dar lungimea necesară afișării este mai mare, se transmite numărul de octeți necesari; iar dacă lungimea necesară afișării este mai mică, se transmite numărul de octeți necesari, iar restul se completează cu caracterul de umplere.



```
#include <iostream.h>
int main()
{
    int i=123, j=456;
    double x=1234.567890123456, y=5678;
    cout.width(5); cout<<i<< ' '<<j<<endl;           //123 456
    cout<<i;           //toate metodele sunt persistente, cu
    excepția metodei width
    cout.width(5); cout<<j<<endl;           //123 456
    cout.setf(ios::showpos);                  //afis. +
    cout<<i<<' '<<j<<endl;           //+123 +456
    cout.setf(ios::hex, ios::basefield);
    cout.width(20); cout.precision(15); cout.fill('*');
    cout.setf(ios::showpos | ios::showpoint);
    cout<<x<<' '<<y<<endl;
    //***+1234.56789012346 +5678.000000000000
    cout<<i<<' '<<j<<endl;           //7b 1c8
    cout.setf(ios::dec, ios::basefield);
    cout.unsetf(ios::showpos); cout.fill(' ');
    cout<<i<<' '<<j<<endl;           //123 456
    cout<<x<<' '<<y<<endl;
    //1234.56789012346 5678.000000000000
}
```

**Observație:** Toate metodele sunt persistente, cu excepția metodei *width* care este valabilă numai pentru operația următoare, după care se setează la 0.

### 7.3. Metodele clasei istream

Clasa *istream* este derivată din clasa *ios*: *ios:istream*.

#### ➤ Supradefinirea operatorului de extracție >>

Operatorul de extracție din *stream-ul* (fluxul) de intrare este supraîncărcat printr-o *funcție membră*, pentru toate tipurile de bază. El are rolul de a extrage din fluxul de intrare caracterele necesare pentru a obține o valoare dintr-un tip de bază.

***istream & operator >> (&tip\_de\_bază);***

Primul argument este implicit (*this*): clasa care îl apelează. Al doilea argument este o referință către un *tip\_de\_bază*. Operatorul poate fi supraîncărcat printr-o *funcție prietenă* (vezi capitolul 11), pentru a extrage din fluxul de intrare informațiile despre un obiect dintr-un tip definit de utilizator (clasă):

***friend istream & operator >>(istream &, clasa &);***

#### ➤ Metoda **get**

***istream & get (char &);***

Metoda extrage din fluxul de intrare un caracter și îl memorează în variabila referință, transmisă ca parametru. Întoarce o referință către *istream*.

**Exemplu:** `char c; cin.get(c);`

Metoda *get* este supraîncărcată și astfel:

***int get ( );***

Extrage din fluxul de intrare un singur caracter și îl returnează sub forma unui întreg. Această metodă este utilizată, în special, la testarea sfârșitului de fișier (EOF, -1).

Metoda *get* este supraîncărcată și astfel:

***istream & get (char \* sir, int lungime, char delimitator = '\n');***

Se extrage din fluxul de date de intrare un sir de caractere (char \* sir = pointer către sir), cu lungimea maximă specificată prin argumentul lungime, până la întâlnirea delimitatorului (delimitator nu se extrage din fluxul de date de intrare). Rezultatul se depune în variabila *sir*.

➤ Metoda **getline**

Metoda determină preluarea din fluxul de intrare a unui sir de caractere, terminat cu un caracter cunoscut.

**istream & getline (char \* sir, int lungime, char delimitator = EOF);**

Acționează asemănător cu metoda get supraîncărcată prin ultima formă, cu deosebirea că din fluxul de intrare se extrage și delimitatorul. Delimitatorul nu se introduce în *sir*.

**Exemplu:** char sir[50]; cin.get (sir, 50, '\n'); // sau cin.get(sir, 50);

Din fluxul de date de intrare se extrag caracterele până la sfârșit de linie, cel mult 50 de caractere. Extragerea caracterelor din fluxul de intrare se termină fie la întâlnirea terminatorului, fie atunci când s-a citit un număr de caractere egal cu *lungime-1*.

➤ Metoda **gcount** returnează un întreg care reprezintă numărul efectiv de caractere preluat prin **getline**.

**int gcount();**

➤ Metoda **read** extrage din fluxul de intrare un sir de caractere (octeți) de lungime impusă și-l depozitează în zona de memorie *sir*.

**istream & read (char \* sir, int lungime);**

**Exemplu:** char t[30]; cin.read(t, 20);

➤ Metoda **ignore** este utilizată la golirea zonei tampon a stream-ului de intrare.

**istream & ignore (int lungime, char delimitator = ' ');**

Se extrag din fluxul de intrare caracterele până la *delimitator*, dar nu mai multe decât numărul indicat de parametru *lungime*. Caracterele extrase sunt *eliminate*, nu sunt memorate.

- Metoda **peck** furnizează primul caracter din fluxul de intrare, fără a-l extrage însă din flux. Caracterul va fi primul caracter extras la următoarea citire.

**int peck( );**

**Exemplu:** c = cin.peck ( );

- Metoda **putback** inserează în fluxul de intrare caracterul trimis ca argument.

**istream & putback(char &);**

**Observații:**

- 1) int a; cin.get (a)  $\Leftrightarrow$  cin >> a;
- 2) Metodele pot fi folosite în serie: **Exemplu:** cin.get(a).get(b).get (c);

## 7.4. Metodele clasei ostream

- Supradefinirea operatorului de inserție <<

Operatorul de inserție în fluxul de ieșire este supraîncărcat printr-o *funcție membră* pentru toate tipurile de bază. El are rolul de a introduce (insera) în fluxul de ieșire caracterele necesare pentru a afișa o valoare dintr-un tip de bază.

**ostream & operator << (tip\_de\_bază);**

Primul argument este implicit (this). Al doilea argument este o expresie de tip de bază.

Operatorul poate fi supraîncărcat printr-o *funcție prietenă*, pentru a insera în fluxul de ieșire informațiile despre un obiect dintr-un tip definit de utilizator (clasă):

**friend ostream & operator << (ostream &, clasa &);**

- Metoda **put** inserează în fluxul de date de ieșire caracterul transmis ca parametru.

**ostream put (char);**

### Exemplu:

```
char c='S'; cout . put ( c ); // echivalent cu cout << c;  
char c1='A',c2='B',c3='C';cout.put(c1).put(c2).put(c3);  
// echivalent cu: cout.put(c1);cout.put(c2); cout.put(c3);  
// echivalent cu cout<<c1<<c2<<c3;
```

- Metoda **write** inserează în fluxul de date de ieșire un număr de caractere, de lungime impusă, existente în zona tablou.

**ostream & write (char \* tablou, int lungime);**

**Exemplu:** char t[ ]="Bună ziua!\n"; cout.write(t, 4);

//Inserează în fluxul de ieșire 4 caractere, începând de la adresa t.

**Exercițiu:** Se ilustrează formatarea unui flux de intrare/ieșire atât prin funcții membre, cât și cu ajutorul manipulatorilor (cu sau fără parametri).



```
#include <iostream.h>  
#include <iomanip.h>  
int main()  
{  
    int i=123; char car, mesaj[]=" Apasă tastă Enter\n";  
    cout<<setw(5)<<resetiosflags(ios::internal |ios::right);  
    cout<<setiosflags(ios::left)<<setfill('#')<<i<<endl;  
    cout<<setw(5)<<setiosflags(ios::showpos)<<setfill(''  
)<<i<<endl;  
    cout<<setw(5)<<resetiosflags(ios::left)<<setfill('0')<<set  
iosflags(ios::right);  
    cout<<i<<endl;  
    cout<<"\n hexagesimal:"<<setw(5)<<hex<<i<<endl;  
    cout<<"\n octal:"<<setw(5)<<oct<<i<<endl;  
    cout<<"\n zecimal:"<<setw(5)<<dec<<i<<endl;  
    cout<<mesaj; cin.get(car);  
    double d=123.456789012345678901234567890123456789;  
    cout<<"Afisare d cu precizia implicită:"<<d<<endl;  
    cout<<"Afisare d cu lung 25 si precizie  
15:"<<setw(25)<<setprecision(15)<<d<<endl;  
    cout<<"Schimbare caracter umplere  
&:"<<setw(25)<<setfill('&')<<d<<endl;  
    cout<<endl;  
}
```

## 7.5. Manipulatori creați de utilizator

Manipulatorii sunt funcții speciale care pot fi utilizate alături de operatorii de inserție/extracție pentru a forma datele de ieșire/intrare. Pe lângă manipulatorii predefiniți, utilizatorul își poate crea manipulatori proprii.

### Crearea manipulatorilor fără parametri

```
ostream &nume_manipulator (ostream &);           //pentru un flux de ieșire  
istream &nume_manipulator (istream &);           //pentru un flux de intrare
```

### Crearea manipulatorilor cu parametri

Crearea manipulatorilor fără parametri necesită folosirea a două funcții:

```
ostream & nume_manipulator (ostream &, int);  
omanip <int> nume_manipulator (int n) // funcție şablon (template) cu  
parametru de tip int  
{return omanip <int> (nume_manipulator, n);}
```

Deci, manipulatorii sunt funcții ale căror corpuri sunt stabilite de utilizator.

**Exemplu:** În exemplul următor se definesc manipulatorii indentare (indentare = înaintea liniei propriu-zise, se lasă un număr de spații albe), convhex (realizează conversia din zecimal în hexa), init (initializează lungimea câmpului la 10 caractere, stabileste precizia la 4 și caracterul de umplere \$).



```
#include <iostream.h>
#include <iomanip.h>
ostream &convhex (ostream &ies)
{
    ies.setf(ios::hex,
    ios::basefield);
    ies.setf(ios::showbase);
    return ies;
}
ostream &indentare (ostream &ies, int nr_spații)
{
    for (int i=0; i<nr_spatii; i++)
        ies<<' ';
    return ies;
}
ostream &init (ostream &ies)
{
    ies.width(10);
    stream.precision(4);
    ies.fill('$');
    return ies;
}
omanip <int> indentare (int nr_spații)
{
    return omanip <int> (indentare, nr_spații);
}
int main()
{
    cout<<712<<' '<<convhex<<712<<endl;
    cout<<indentare(10)<<"Linia1
text\n"<<indentare(5)<<"Linia2 text\n";
    cout init<<123.123456;
}
```

## 7.6. Fluxuri de date pentru fișiere

Un flux de intrare/ieșire poate fi asociat unui fișier.

Pentru a asocia un **flux de ieșire** unui fisier, este necesară **crearea** unui **obiect** de tipul **ofstream**. Clasa **ofstream** este *derivată* din clasele **ostream** (moștenește metodele care permit inserarea informațiilor într-un flux

de ieșire) și **fstreambase** (moștenește operațiile privitoare la asocierea fișierelor).

Pentru a asocia un **flux de intrare** unui fișier, este necesară crearea unui obiect de tipul **ifstream**. Clasa ifstream este *derivată* din clasele **istream** (moștenește metodele care permit extragerea informațiilor dintr-un flux de intrare) și **fstreambase** (moștenește operațiile privitoare la asocierea fișierelor).

Pentru crearea unor obiecte din clasele ifstream sau ofstream se impune includerea headere-lor **iostream.h** și **fstream.h**.

**Constructorii clasei ofstream sunt:**

**ofstream();**

Fluxul nu este asociat nici unui fișier

**ofstream(const char \*nume\_fisier, int mod\_acces=ios::out);**

Parametrii constructorului sunt numele fișierului și modul de acces la acesta (scriere, fișier de ieșire).

**ofstream (int fd, char \*buffer, int lung\_buffer);**

Parametrii sunt numărul intern al fisierului (fd), zona tampon de intrări/ieșiri (buffer) și lungimea zonei tampon (lung\_buffer).

**Constructorii clasei ifstream sunt:**

**ifstream();**

Fluxul nu este asociat nici unui fișier

**ifstream(const char \*nume\_fisier, int mod\_acces=ios::in);**

Parametrii constructorului sunt numele fișierului și modul de acces la acesta (citire, fișier de intrare).

**ifstream (int fd, char \*buffer, int lung\_buffer);**

Parametrii sunt numărul intern al fișierului (fd), zona tampon de intrări/ieșiri (buffer) și lungimea zonei tampon (lung\_buffer).

### **Exemplu:**

Se declară obiectul numit fis\_ies, de tipul ofstream, obiect asociat unui fișier cu numele DATE.txt, deschis apoi pentru ieșire (scriere). Scrierea în fișierul asociat obiectului se face printr-un flux care beneficiază de toate "facilitățile" clasei ostream.

```
ofstream fis_ies("DATE.txt", ios::out); //sau: ofstream fis_ies("DATE.txt");
```

Scrierea în fișierul DATE.txt: fis\_ies<< . . . << . . . ;

Pentru scriere formatată sau scriere binară în fișierul asociat:

```
fis_ies.write(. . . );
```

Pentru examinarea stării fluxului de eroare corespunzător: if (fis\_ies) . . . ;

Se declară obiectul numit fis\_intr, de tipul ifstream, obiect asociat unui fișier cu numele NOU.txt, deschis apoi pentru intrare (citire). Citirea din fișierul asociat obiectului se face printr-un flux care beneficiază de toate "facilitățile" clasei istream.

```
ifstream fis_intr("DATE.dat", ios::in); //sau: ifstream fis_intr("NOU.txt");
```

Citirea în fișierul NOU.txt: fis\_intr>> . . . >> . . . ;

Pentru citire formatată sau citire binară din fișierul asociat: fis\_intr.read(. . . );

Pentru examinarea stării fluxului de eroare corespunzător: if (fis\_intr) . . . ;

***Observații:*** Conectarea (asocierea) unui flux unui fișier presupune:

- Fie existența a două tipuri de obiecte: un flux și un fișier;
- Fie declararea unui flux care va fi asociat ulterior unui fișier.

Clasa **fstream** moștenește atât clasele ifstream, cât și ofstream. Ea permite accesul în citire/scriere. Constructorul cu parametri al clasei fstream:

```
fstream(char *nume_fisier,int mod_deschid,int protec=filebuf::openprot);
```

Modul de deschidere (mod\_deschid) a unui fișier poate fi:

ios::in            fișier de intrare

ios::out          fișier de ieșire

ios::ate          după deschiderea fișierului, poziția curentă este sfârșitul acestuia

<b>ios::app</b>	mod append (de scriere la sfârșitul unui fișier existent)
<b>ios::trunc</b>	dacă fișierul există, datele existente se pierd
<b>ios::nocreate</b>	dacă fișierul asociat nu există, nu va fi creat decât la scriere
<b>ios::binary</b>	fișier binar
<b>ios::noreplace</b>	fișierul nu trebuie să existe

Modul de deschidere este definit printr-un cuvânt de stare, în care fiecare bit are o semnificație particulară. Pentru setarea (activarea) mai multor biți, se folosește operatorul | (sau logic, pe bit), ca în exemplu.

```
class ios{
//.....
public:
enum open_mode{
    in=0x01, out=0x02, ate=0x04,
    app=0x08, trunc=0x10, nocreate=0x20,
    noreplace=0x40, binary=0x80
};
};
```

**Exemplu:** ifstream f1("DATE.txt", ios::in|ios::nocreate);  
 fstream f2("pers.dat", ios::in|ios::out);

#### ➤ Metoda open

Prelucrarea unui fișier începe cu deschiderea acestuia, care se poate realiza:

- La instanțierea obiectelor din clasele ifstream, ofstream sau fstream, cu ajutorul constructorului cu parametri (constructorul apelează automat funcția open);

- Prin apelul explicit al funcției open.

Metoda open este definită în clasa **fstreambase** și este supraîncărcată în funcțiile derivate din aceasta. Ea are aceeași parametri ca și constructorul clasei și prototipul:

```
void open (char *nume_fisier, int mod_deschidere, int protecție);
```

- Metoda **close** realizează închiderea unui fișier:

```
void close();
```

**Exercițiu:** În exemplul următor se asociază fluxului de ieșire (obiectului `fis_ies`) fișierul numit `text.txt`. Se testează cuvântul de stare de eroare, iar dacă nu au fost probleme la deschidere, în fișier se scrie un text (2 linii), apoi, o serie de constante numerice, formatare. Se închide fișierul. Același fișier este asociat unui flux de intrare (pentru citire). Textul (scris anterior) este citit în variabila `șir`, apoi este afișat. Se citesc din fișier și apoi sunt afișate constantele numerice. Se închide fișierul.



```
#include <fstream.h>
#include <iomanip.h>
int main()
{
    ofstream fis_ies("text.txt");           //deschidere fișier
    text.txt
    if (!fis_ies){
        cout<<"Eroare la deschiderea fișierului!\n";
        return 1; }
    fis_ies<<"Scrierea unui text \n în fișier\n";
    fis_ies<<100<<hex<<setw(10)<<100<<setw(20)<<setprecision(12);
    cout<<123.456789012356789<<endl;
    fis_ies.close();                      //închiderea fișierului
    ifstream fis_intr("text.txt"); //deschiderea fis. ptr. citire
    if (!fis_intr)
    {
        cout<<"Eroare la deschiderea fis. ptr. citire!\n";
        return 1; }
    char sir[100];
    for (int k=0; k<9; k++)
    {
        fis_intr>>sir; cout<<sir<<'\n';
        cout<<endl;
    int i, j;
    double u;
    fis_intr>>i>>hex>>j>>u;
    cout<<i<<' '<<j<<' '<<u<<endl;
    fis_intr.close();
    return 0;
}
```

### *Observații:*

Deschiderea și închiderea fișierului s-ar fi putut realiza și astfel:

```
fstream fis_ies; fis_ies.open("text.txt", ios::out); fis_ies.close();  
fstream fis_in; fis_in.open("text.txt", ios::in); fis_in.close();
```

**Exemplu:** Să se scrie un program care concatenează două fișiere, depunând rezultatul în al treilea fișier. Se crează trei fluxuri, f1, f2 și dest, care sunt atașate fișierelor corespunzătoare (prin metoda open). Copierea în fișierul destinație se realizează prin folosirea metodelor get, put și funcția copy. Metoda close întrerupe legătura logică dintre fluxuri și fișiere.



```
#include <iostream.h>  
#include <process.h>  
#include <fstream.h>  
void semn_er(char *c)  
{ cerr<<"\n\n Eroare la deschiderea fisierului "<<c<<endl;  
    exit(1); }  
void copiere(ofstream &dest, ifstream &sursa)  
{     char C;  
        while (dest && sursa.get(C))  
            dest.put(C);  
    }  
int main()  
{  
    char nume_sursa1[14], nume_sursa2[14], destinatie[14];  
    ifstream f1, f2;  
    ofstream dest;  
    cout<<"Numele celor 3 fisiere:"<<"Destinatie Sursa1  
Sursa2\n";  
    cin.read(destinatie,13);  
    cin.read(nume_sursa1,13);  
    cin.read(nume_sursa2,13);  
    destinatie[13]=nume_sursa1[13]=nume_sursa2[13]='\0';  
    f1.open(nume_sursa1, ios::nocreate);  
    if (!f1) semn_er(nume_sursa1); //utiliz operator ! redefinit  
    f2.open(nume_sursa2, ios::nocreate);  
    if (!f2) semn_er(nume_sursa2); //utiliz operator ! redefinit  
    dest.open(destinatie);  
    if (!dest) semn_er(destinatie); //utiliz operator ! redefinit  
    copiere (dest, f1);  
    copiere(dest, f2);  
    f1.close();  
    f2.close();  
    dest.close();  
}
```

**Exercițiu:** Se ilustrează folosirea fluxurilor pentru operații de citire/scriere în fișiere text.



```
#include <fstream.h>
#include <iostream.h>
int main()
{
    ofstream fisierout("nou.txt");      // Deschide fisierul text nou.txt.
    fisierout << " Teste fisiere "; //Scrie un text in fisier
    fisierout.close(); // Inchide fisierul
    // Deschide un alt fisier text si scrie in acesta niste numere (numerele sunt separate prin spatii)
    fisierout.open("numere.txt");
    fisierout <<15<<" "<<42<<" "<<1;
    fisierout.close();
    // Deschide fisierul nou.txt pentru citire.
    ifstream fisierin;
    fisierin.open("nou.txt");
    char p[50]; // Stabileste o zona de memorie folosita pentru citirea textului.
    fisierin >>p; // Citeste si afiseaza primele doua cuvinte din fisier.
    cout <<p<<" ";
    fisierin >>p;
    cout <<p<<endl;
    fisierin.close(); // Inchide fisierul.
    int numar;
    fisierin.open("numere.c");
    // Deschide fisierul text numere.c pentru citirea numerelor intregi
    // Citeste numere din fisier pana cand ajunge la sfarsitul fisierului.
    while (!fisierin.eof())
    {
        fisierin >> numar; // Citeste un numar intreg.
        if (!fisierin.eof()) cout <<numar<<endl;
        // Daca nu a ajuns la sfarsitul fisierului afiseaza numarul.
    }
    fisierin.close(); // Inchide fisierul.
    // se citeste textul din fisierul nou.txt cuvant cu cuvant.
    fisierin.open("nou.txt");
    while (!fisierin.eof())
        // Citeste cuvinte din fisier pana ajunge la sfarsitul fisierului.
    {
        // Citeste cuvant cu cuvant.
        fisierin >>p;
        cout <<p<<endl;
    }
}
```

## 7.7. Fișiere binare

**Fișierele binare** reprezintă o succesiune de octeți, asupra cărora la intrare sau la ieșire nu se realizează nici o conversie. Ele sunt prelucrate binar, spre deosebire de exemplul anterior, în care prelucrarea se realiza în mod text (un număr în format intern este reprezentat binar, dar în format extern el apare ca un sir de caractere. Un sir în formatul extern are terminatorul '\n', iar în format intern are terminatorul '\0'). Deoarece nu există un terminator de linie, trebuie specificată lungimea înregistrării. Metodele write, read nu realizează nici o conversie.

- Metoda **write** scrie în fișier un număr de n octeți:

```
ostream &write(const char * tab, int n);
```

- Metoda **read** citește din fișier n octeți.

```
istream &read (const char * tab, int n);
```

Limbajul C++ oferă (ca și limbajul C) posibilitatea de acces direct într-un fișier conectat la un flux de date. După fiecare operație de citire (extragere din fluxul de intrare) sau citire (inserție în fluxul de ieșire), pointerul care indică poziția curentă în fluxul de date este incrementat cu un număr egal cu numărul de octeți transferați. În această situație se realizează un acces secvențial (ca în cazurile precedente).

Clasele **ifstream** și **ofstream** au ca metode funcțiile **seekg** (membră a clasei **istream**), respectiv **seekp** (membră a clasei **ostream**) care permit modificarea valorii pointerului.

```
istream & seekg(long);
```

```
istream & seekg(long, seek_dir);
```

```
ostream & seekp(long);
```

```
ostream & seekp(long, seek_dir);
```

Ambele metode primesc doi parametri:

- un întreg reprezentând deplasarea pointerului în raport cu baza (referința) precizată de al doilea parametru;
- o constantă întreagă care precizează baza. Valorile constantei sunt definite în clasa ios:



```
class ios{
    ...
public:
    enum seek_dir{
        beg=0, deplasare față de începutul fișierului (implicit)
        cur=1, deplasare față de poziția curentă
        end=2 deplasare față de sfârșitul fișierului
    };
};
```

Pentru aflarea valorii pointerului, clasa ifstream are metoda **tellg**, iar clasa ofstream are metoda **tellp**, cu prototipurile:

```
long istream::tellg();
long ostream::tellp();
```

**Exemplu:** Se crează ierarhia de clase din figura 7.5., în care se implementează clasa student și clasa proprie fisier\_stud, clasă derivată din fstream. În același fișier, binar, se realizează și scrierea și citirea. Datorită posibilității de acces direct, se pot modifica informațiile referitoare la studentul înregistrat în fișier pe poziția k.

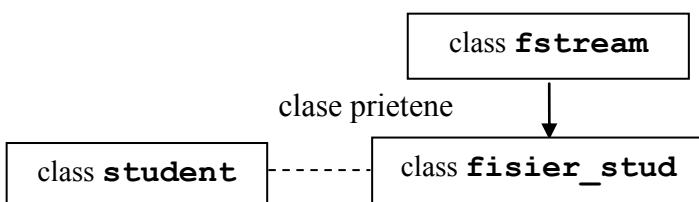


Figura 7.5. Ierarhia de clase



```
#include <fstream.h>
#include <iomanip.h>
class student
{
    char nume[20];
    int grupa, note[3];
public:
    void citire_date();
    friend ostream &operator<<(ostream &, const student &);
    friend class fisier_stud;
};

class fisier_stud:public fstream //fișier binar cu obiecte din cls
student
{
public:
    fisier_stud(){;};
    fisier_stud(const char*num_f,int mod,int
    protecție=filebuf::openprot):fstream(num_f,mod
    |ios::binary,protecție){ }
void open(const char *num_f, int mod, int
protectie=filebuf::openprot)
{
    fstream::open(num_f,mod|ios::binary,protectie);}
//apelul met. open din cls. fstream
    int citeste_f(student &);
    int scrie_f (const student &);
};

void student::citire_date()
{
    int gr, OK;
    for (int k=0; k<21; k++)  nume[k]=0;
    cin.ignore(1000, '\n');
    cout<<"Numele studentului:";
    cin.get(nume, 21);
    cin.ignore(1000, '\n');
    do{
        cout<<"Grupa:";
        cin>>gr;
        OK=(cin && gr>0 && gr<8000);
        if (!OK)
        {
            cout<<"Eroare. Repetați introducerea!\n";
            cin.clear();
        }
        else {      grupa=gr; cin.ignore(1000, '\n');      }
    }while (!OK);
```

```

do{
    cout<<"Scrieți date în fișier [D|N] ?";
    rasp=toupper(cin.get());cin.ignore(1000, '\n');
} while (răsp!='D' && răsp!='N');
if (răsp == 'D')
{ fs.scrie_f(s); cout<<"Poz. în fișier: "<<fs.tellp()<<endl; }
do{
    cout<<"Continuați                                introducerea
[D|N]?" ;răsp=toupper(cin.get()); cin.ignore(1000, '\n');
} while (răsp!='D' && răsp!='N');
if (răsp=='N')           gata=1;
}
fs.close();
//citire
fs.open(Nume_Fis, ios::in);
if (!fs){
    cout<<"Eroare la deschiderea fiș-lui "<<Nume_Fis<<
ptr. citire\n"; return 1;
cout<<"Poz. în fiș la deschidere:"<<fs.tellg()<<endl;
cout<<"Date citite din fișier:\n"; int k=0;
while (!fs.eof())
{
    fs.citește_f(s);
    if (!fs && !fs.eof()) { cout<<"Eroare la citire\n";
return 1;
    if (!fs.eof()) { cout<<s<<endl; cout<<"Poz. în
fișier:"<<fs.tellg();
cout<<endl;k++; }
}
cout<<"S-au citit din fișier:"<<k<<" studenți\n";
fs.close();

fs.open(Nume_Fis, ios::in | ios::out); //deschidere fisier
actualizare (intr/ies)
if (!fs){
    cout<<"Eroare la deschidere fis. "<<Nume_Fis<<" ptr.
citire/ scriere\n"; return 1;
}
cout<<"Dați numărul stud-lui pentru care vreți să înlocuiți
datele:"; cin>>k;
cin.ignore(1000, '\n');
fs.seekg(k*sizeof(stud)); s.citire_date(); fs.scrie_f(s); fs.seekg(0);
k=0;
while(!fs.eof()){
    fs.citire_f(s);
    if (!fs && !fs.eof()) { cout<<"Eroare la citirea din
fișier:"<<Nume_Fis<<endl; return 1; }
    if (!fs.eof()){
        cout<<s<<endl; cout<<"Poz. în fișier:"<<fs.tellg()<<endl; k++; }
}
cout<<"S-au gasit în fișier:"<<k<<" studenți\n";
fs.close(); return 0;
}

```

# Functii din biblioteca standard

**I**n acest capitol sunt descrise functii care rezolvă probleme legate de alocarea dinamică a memoriei, sortare și căutare, clasificare, operații cu blocuri de memorie și siruri de caractere, functii matematice.

## 8.1. Alocarea dinamică a memoriei

### Nume

**calloc, malloc, realloc** - alocă memoria în mod dinamic  
**free** - elibereză memoria alocată în mod dinamic

### Declarație



```
void *calloc(unsigned nel, unsigned size);
void *malloc(unsigned size);
void *realloc(void *ptr, unsigned size);
void free(void *ptr);
```

### Descriere

Funcția **calloc** alocă memorie pentru un tablou de nel elemente, fiecare de mărime size octeți și returnează un pointer la memoria alocată. Conținutul memoriei este pus la zero.

Funcția **malloc** alocă size octeți și returnează un pointer la memoria alocată. Conținutul memoriei nu este șters.

Funcția **free** eliberează spațiul de memorie indicat de ptr, care trebuie să fi fost returnat de un apel anterior malloc, calloc sau realloc. În caz contrar, sau dacă a existat deja un apel anterior free(ptr), comportamentul programului este imprevizibil.

Funcția **realloc** schimbă mărimea blocului de memorie indicat de ptr la size octeți. Conținutul rămîne neschimbat la mărimea minimă dintre mărimea veche și cea nouă; noul spațiu de memorie care este eventual alocat este neinitializat. Dacă ptr este NULL apelul este echivalent cu malloc(size); dacă size este egal cu zero apelul este echivalent cu free(ptr). Cu excepția cazului cînd ptr este NULL, acesta trebuie să fi fost returnat de un apel precedent malloc, calloc sau realloc.

#### Valori returnate

Pentru **calloc** și **malloc** valoarea returnată este un pointer la memoria alocată, care este aliniată în mod corespunzător pentru orice tip de variabile, sau NULL dacă nu există suficientă memorie continuă.

Funcția **free** nu returnează nimic.

Funcția **realloc** returnează un pointer la noua zonă de memorie alocată, care este aliniată în mod corespunzător pentru orice tip de variabile, și poate fi diferită de ptr, sau poate fi NULL dacă nu există suficientă memorie continuă sau dacă valoarea size este egală cu 0. Dacă **realloc** eșuează, blocul original rămîne neatins - nu este nici eliberat nici mutat.

## 8.2. Sortare și căutare

### Nume

**qsort** - sortează un tablou

**bsearch** - căutare binară într-un tablou sortat

## Declarație



```
void qsort(void *base, unsigned nel, unsigned size, int (*comp) (const void *, const void *));
void *bsearch(const void *key, const void *base, unsigned nel, unsigned size, int (*comp)(const void *, const void *));
```

## Descriere

Funcția **qsort** sortează un tablou de *nel* elemente, fiecare de mărime *size*. Argumentul *base* indică spre începutul tabloului.

Elementele tabloului sunt sortate în ordine crescătoare în concordanță cu funcția de comparare referită de *comp*, apelată cu două argumente care indică spre obiectele ce se compară. Funcția de comparare trebuie să returneze un întreg mai mic decât, egal cu, sau mai mare decât zero dacă primul argument este considerat a fi mai mic decât, egal cu, respectiv mai mare decât al doilea. Dacă cele două elemente comparate sunt egale, ordinea în tabloul sortat este nedefinită.

Funcția **bsearch** caută într-un tablou de *nel* elemente, fiecare de mărime *size*, un membru care coincide cu obiectul indicat de *key*. Argumentul *base* indică spre începutul tabloului.

Conținutul tabloului trebuie să fie sortat crescător în concordanță cu funcția de comparare referită de *comp*, apelată cu două argumente care indică spre obiectele ce se compară. Funcția de comparare trebuie să returneze un întreg mai mic decât, egal cu, sau mai mare decât zero dacă primul argument este considerat a fi mai mic decât, egal cu, respectiv mai mare decât al doilea.

## Valoare returnată

Funcția **bsearch** returnează un pointer la un membru al tabloului care coincide cu obiectul indicat de *key*, sau NULL dacă nu se găsește nici un membru. Dacă există mai multe elemente care coincid cu *key*, poate fi returnat oricare element cu această proprietate.

## 8.3. Funcții de clasificare

Nume

**isalnum**, **isalpha**, **isascii**, **iscntrl**, **isdigit**,  
**isgraph**, **islower**, **isprint**, **ispunct**, **isspace**,  
**isupper**, **isxdigit** - rutine de clasificare  
**tolower** - conversie în literă mică  
**toupper** - conversie în literă mare

Declarație



```
#include <ctype.h>
    int isalnum(int c);      int islower(int c);
    int isalpha(int c);     int isprint(int c);
    int isascii(int c);     int ispunct(int c);
    int iscntrl(int c);    int isspace(int c);
    int isdigit(int c);    int isupper(int c);
    int isgraph(int c);    int isxdigit(int c);
    int tolower(int c);    int toupper(int c);
```

Descriere

Primele 12 funcții verifică dacă c, care trebuie să fie o valoare de tip unsigned char sau EOF, se află în una din clasele de caractere enumerate mai sus.

**isalnum**

Verifică dacă c este alfanumeric; este echivalentă cu (isalpha(c) || isdigit(c)).

**isalpha**

Verifică dacă c este alfabetic; este echivalent cu (isupper(c) || islower(c)).

**isascii**

Verifică dacă c este o valoare pe 7 biți din setul de caractere ASCII.

**iscntrl**

Verifică dacă c este un caracter de control.

**isdigit**

Verifică dacă c este o cifră (între 0 și 9).

**isgraph**

Verifică dacă c este un caracter afișabil cu excepția spațiului.

**islower**

Verifică dacă c este o literă mică.

**isprint**

Verifică dacă c este un caracter afișabil inclusiv spațiu.

**ispunct**

Verifică dacă c este un caracter diferit de spațiu și non-alfanumeric.

**isspace**

Verifică dacă c este un spațiu alb.

**isupper**

Verifică dacă c este o literă mare.

**isxdigit**

Verifică dacă c este o cifră hexazecimală din setul 0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F.

**tolower**

Convertește caracterul c, dacă este o literă, la literă mică corespunzătoare.

### **toupper**

Convertește caracterul c, dacă este o literă, la litera mare corespunzătoare.

Valoare returnată

Valoarea returnată de funcțiile is... este nenulă dacă caracterul c se află în clasa testată, și zero în caz contrar.

Valoarea returnată de funcțiile to... este litera convertită dacă caracterul c este o literă, și nedefinită în caz contrar.

## **8.4. Operații cu blocuri de memorie**

Pentru majoritatea funcțiilor din această categorie compilatorul expandează codul acestora folosind instrucțiuni pe șiruri de caractere. Declarațiile acestor funcții se obțin cu

```
#include <string.h>
```

Nume

**memcpy** - copiază o zonă de memorie

Declarație



```
void *memcpy(void *dest, const void *src, unsigned n);  
void *memmove(void *dest, const void *src, unsigned n);
```

Descriere

Funcția memcpy copiază n octeți din zona de memorie src în zona de memorie dest. Zonele de memorie nu trebuie să se suprapună. Dacă există acest risc se utilizează memmove.

Valoare returnată

Funcțiile returnează un pointer la dest.

Nume

**memcmp** - compară două zone de memorie

Declarație



**int memcmp(const void \*s1, const void \*s2, unsigned n);**

Descriere

Funcția memcmp compară primii n octeți ai zonelor de memorie s1 și s2.

Valoare returnată

Returnează un întreg mai mic decât, egal cu, sau mai mare decât zero dacă s1 este mai mic decât, coincide, respectiv este mai mare decât s2.

Nume

**memset** - umple o zonă de memorie cu o constantă pe un octet

Declarație



**void \*memset(void \*s, int c, unsigned n);**

Descriere

Funcția memset umple primii n octeți ai zonei de memorie indicată de s cu constanta c pe un octet.

Valoare returnată

Funcția returnează un pointer la zona de memorie s.

Nume

**memchr** - caută în memorie un caracter

Declarație



**void \*memchr(const void \*s, int c, unsigned n);**

## Descriere

Funcția memchr caută caracterul c în primii n octeți de memorie indicați de s. Căutarea se oprește la primul octet care are valoarea c (interpretată ca unsigned char).

## Valoare returnată

Funcția returnează un pointer la octetul găsit sau NULL dacă valoarea nu există în zona de memorie.

## 8.5. Operări cu șiruri de caractere

Pentru majoritatea funcțiilor din această categorie compilatorul expandează codul acestora folosind instrucțiuni pe șiruri de caractere. Declarațiile acestor funcții se obțin cu

```
#include <string.h>
```

### Nume

**strlen** - calculează lungimea unui șir

### Declarație



```
unsigned strlen(const char *s);
```

### Descriere

Funcția strlen calculează lungimea șirului s, fără a include caracterul terminator null.

## Valoare returnată

Funcția returnează numărul de caractere din s.

### Nume

**strcpy, strncpy** - copiază un șir de caractere

## Declarație



```
char *strcpy(char *dest, const char *src);  
char *strncpy(char *dest, const char *src, unsigned n);
```

## Descriere

Funcția strcpy copiază șirul indicat de src (inclusiv caracterul terminator null) în zona indicată de dest. Șirurile nu trebuie să se suprapună, și în plus zona dest trebuie să fie suficient de mare pentru a primi copia.

Funcția strncpy este similară, cu excepția faptului că nu se copiază mai mult de n octeți din src. Astfel, dacă caracterul terminator null nu se află în primii n octeți din src, rezultatul nu va fi terminat cu null. În cazul în care lungimea lui src este mai mică decât n, restul octețiilor din dest primesc valoarea null.

## Valoare returnată

Funcțiile returnează un pointer la șirul dest.

## Nume

**strdup** - duplică un șir

## Declarație

```
char *strdup(const char *s);
```

## Descriere

Funcția strdup returnează un pointer la un nou șir care este un duplicat al șirului s. Memoria pentru noul șir se obține cu malloc, și poate fi eliberată cu free.

## Valoare returnată

Funcția returnează un pointer la șirul duplicat, sau NULL dacă nu există memorie suficientă disponibilă.

## Nume

**strcat, strncat** - concatenează două șiruri

### Declarație

```
char *strcat(char *dest, const char *src);  
char *strncat(char *dest, const char *src, unsigned n);
```

### Descriere

Funcția strcat adaugă șirul src la șirul dest suprascriind caracterul null de la sfîrșitul lui dest, și la sfîrșit adaugă un caracter terminator null. Șirurile nu trebuie să se suprapună, și în plus șirul dest trebuie să aibă suficient spațiu pentru a păstra rezultatul.

Funcția strncat este similară, cu excepția faptului că numai primele n caractere din src se adaugă la dest.

### Valoare returnată

Funcțiile returnează un pointer la șirul rezultat dest.

### Nume

**strcmp** - compară două șiruri de caractere

### Declarație

```
int strcmp(const char *s1, const char *s2);
```

### Descriere

Funcția strcmp compară cele două șiruri s1 și s2.

### Valoare returnată

Funcția returnează un întreg mai mic decât, egal cu, sau mai mare decât zero dacă s1 este mai mic decât, coincide, respectiv este mai mare decât s2.

### Nume

**strchr, strrchr** - localizează un caracter

### Declarație

```
char *strchr(const char *s, int c);  
char * strrchr(const char *s, int c);
```

### Descriere

Funcția strchr returnează un pointer la prima apariție a caracterului c în sirul s.

Funcția strrchr returnează un pointer la ultima apariție a caracterului c în sirul s.

### Valoare returnată

Funcțiile returnează un pointer la caracterul găsit sau NULL dacă valoarea nu a fost găsită.

### Nume

**strstr** - localizează un subșir

### Declarație

```
char *strstr(const char *sir, const char *subs);
```

### Descriere

Funcția strstr găsește prima apariție a subșirului subs în sirul sir. Caracterul terminator null nu este luat în considerare.

### Valoare returnată

Funcția returnează un pointer la începutul subșirului, sau NULL dacă subșirul nu este găsit.

### Nume

**strspn, strcspn** - caută un set de caractere într-un sir

### Declarație

```
unsigned strspn(const char *s, const char *acc);
```

```
unsigned strcspn(const char *s, const char *rej);
```

#### Descriere

Funcția strspn calculează lungimea segmentului inițial din s format în întregime numai cu caractere din acc.

Funcția strcspn calculează lungimea segmentului inițial din s format în întregime numai cu caractere care nu se găsesc în rej.

#### Valori returnate

Funcția strspn returnează poziția primului caracter din s care nu se află în acc.

Funcția strcspn returnează poziția primului caracter din s care se află în rej.

## 8.6. Biblioteca matematică

1) Funcțiile din prima categorie sînt descrise în <stdlib.h>.

#### Nume

**rand, srand** - generarea numerelor pseudo-aleatoare

#### Declarație

```
int rand(void);
void srand(unsigned int seed);
```

#### Descriere

Funcția rand returnează un întreg pseudo-aleator între 0 și RAND\_MAX (pentru majoritatea mediilor de programare C această constantă este egală cu valoarea maximă cu semn reprezentabilă pe un cuvînt al sistemului de calcul).

Funcția `srand` inițializează generatorul cu valoarea `seed` pentru o nouă secvență de valori întregi pseudo-aleatoare care vor fi returnate de `rand`. Aceste secvențe se repetă dacă `srand` se apelează cu aceeași valoare `seed`.

Se obișnuiește ca generatorul să fie inițializat cu o valoare dată de ceasul sistemului de calcul, ca în exemplul de mai jos:

```
#include <time.h>
srand(time(NULL));
```

Valoare returnată

Funcția `rand` returnează o valoare între 0 și `RAND_MAX`.

Observație

În lucrarea *Numerical Recipes in C: The Art of Scientific Computing* - William H Press, Brian P Flannery, Saul A Teukolsky, William T Vetterling / New York: Cambridge University Press, 1990 (1st ed, p 207), se face următorul comentariu:

"Dacă dorîți să generați o valoare aleatoare întreagă între 1 și 10, se recomandă să folosiți secvența

```
j=1+(int)(10.0*rand()/(RAND_MAX+1.0));
```

și nu o secvență de tipul

```
j=1+(int)(1000000.0*rand())%10;
```

care folosește biții de rang inferior."

Tot în fișierul `<stdlib.h>` sunt descrise și următoarele funcții:

<code>int abs(int i);</code>	valoare absolută
<code>long labs(long i);</code>	valoare absolută
<code>int atoi(char *s);</code>	conversie din ASCII în întreg
<code>long atol(char *s);</code>	conversie din ASCII în întreg lung
<code>double atof(char *s);</code>	conversie din ASCII în dublă precizie

2) Funcțiile din a doua categorie sunt descrise în fișierul `<math.h>`.

double fabs(double x);	valoare absolută
double floor(double x);	parte întreagă inferioară
double ceil(double x);	parte întreagă superioară
double sqrt(double x);	$\sqrt{x}$
double sin(double x);	$\sin(x)$
double cos(double x);	$\cos(x)$
double tan(double x);	$\operatorname{tg}(x)$
double asin(double x);	$\arcsin(x)$
double acos(double x);	$\arccos(x)$
double atan(double x);	$\operatorname{arctg}(x)$ în $[-\pi/2, \pi/2]$
double atan2(double y, double x);	$\operatorname{arctg}(y/x)$ în $[-\pi, \pi]$
double exp(double x);	$e^x$
double log(double x);	$\ln(x)$
double pow(double x, double y);	$x^y$
double sinh(double x);	$\sinh(x)$
double cosh(double x);	$\cosh(x)$
double tanh(double x);	$\operatorname{tgh}(x)$
double ldexp(double x, int e);	$x \cdot 2^e$
double fmod(double x, double y);	x modulo y

Funcția fmod returnează o valoare  $f$  definită astfel:  $x = a \cdot y + f$  și  $a$  este o valoare întreagă (dată de  $x/y$ ) și  $0 \leq |f| < y$ ,  $f$  are semnul lui  $x$ .

Următoarele două funcții returnează două valori: una este valoarea returnată de funcție (de tip double), și cealaltă returnată prin intermediul unui argument de tip pointer la int respectiv double.

```
double frexp(double x, int *e);
```

Funcția frexp desparte valoarea  $x$  în două părți: o parte fracționară normalizată ( $f \in [0.5, 1)$ ) și un exponent  $e$ . Dacă  $x$  este 0 atunci  $f=0$  și  $e=0$ . Valoarea returnată este  $f$ .

```
double modf(double x, double *n);
```

Funcția modf desparte valoarea  $x$  în două părți: o parte fracționară subunitară  $f$  și o parte întreagă  $n$ . Valorile  $f$  și  $n$  au același semn ca și  $x$ . Valoarea returnată este  $f$ .

## 8.7. Programe demonstrative

1) Programul prezentat în continuare generează un sir de n valori întregi aleatoare în intervalul [0, M–1] pe care le depune în tabloul X (alocat dinamic), și apoi le sortează crescător. În continuare se generează k valori întregi aleatoare pe care le caută în tabloul X. Pentru fiecare căutare cu succes se afișează pe terminal valoarea căutată și poziția în tablou.

Valorile n, k și M se iau în această ordine din linia de comandă.



```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
int cmp(const void *A, const void *B)
{
    return *(int *)A-*(int *)B;
}
int main(int ac, int **av)
{
    int *X,*p,M,n,k,i,v;
    if (ac!=4) {
        fputs("Trei argumente!\n",stderr);
        return 1;
    }
    n=atoi(av[1]);
    k=atoi(av[2]);
    M=atoi(av[3]);
    X=(int *)malloc(n*sizeof(int));
    if (!X) return 1;
    srand(time(NULL));
    for (i=0; i<n; i++)
        X[i]=rand()%M;
    qsort(X,n,sizeof(int),cmp);
    for (i=0; i<k; i++)
    {
        v=rand()%M;
        p=(int *)bsearch(&v,X,n,sizeof(int), cmp);
        if (p)
            printf("Val: %d  Pos: %d\n",v,p-X);
    }
    free(X);
    return 0;
}
```

2) Să reluăm al treilea exemplu din capitolul precedent. Se citește un fișier text care conține pe fiecare linie un nume (șir de caractere fără spațiu) și trei valori reale (note). Pentru fiecare linie se calculează media aritmetică a celor trei valori și se determină dacă elementul este admis (fiecare notă este minimum 5) sau respins (cel puțin o notă este sub 5). În final se afișează liniile în ordinea următoare: mai întâi elementele admise în ordinea descrescătoare a mediei, și apoi elementele respinse în ordine alfabetică după nume. Se afișează doar numele, situația (A/R) și media.

În acest exemplu punem în evidență o modalitate comodă de selectare a membrilor unei structuri cu ajutorul macrourilor. Macroul Fld selectează din zona referită de pointerul P membrul f. Deoarece pointerul P (care poate fi argumentul A sau B al funcției comp) referă o zonă de tip void, este necesar mai întâi un *cast* pentru a preciza tipul concret al acesteia. Membrul f poate fi: nm, ar, md, definiți în cadrul structurii de tip StEl.

Deoarece nu știm de la început câte linii are fișierul de intrare, suntem nevoiți să folosim următoarea strategie. La început alocăm pentru tabloul El o zonă care să memoreze NA elemente. Pe măsură ce această zonă se completează, la un moment dat numărul de linii citite coincide cu NA. În acest moment se alocă o zonă nouă care să poată memora un număr mai mare de elemente. Desigur, de fiecare dată se va actualiza mărimea spațiului alocat:



```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#define NA 32
typedef struct {
    char nm[10], ar;
    float na, nb, nc, md;
} StEl;
#define Fld(P,f) ((StEl *)P)->f
int comp(const void *A, const void *B) {
    float w; int d;
    if (d=Fld(A,ar)-Fld(B,ar)) return d;
    if (Fld(A,ar)=='A') {
        w=Fld(B,md)-Fld(A,md);
        if (w>0) return 1;
        if (w<0) return -1;
    }
    return strcmp(Fld(A,nm),Fld(B,nm)); }
```

```

int main(int ac, char **av)
{
    int na,ne,i;
    StEl *El;
    FILE *fi;
    if (ac!=2) {
        fputs("Un argument!\n",stderr);
        return 1;
    }
    fi=fopen(av[1],"rt");
    if (!fi) {
        perror("Eroare la deschidere");
        return 1;
    }
    na=NA; ne=0;
    El=(StEl *)malloc(na*sizeof(StEl));
    while (fscanf(fi,"%s %d %d %d",
        El[ne].nm,&El[ne].na,&El[ne].nb, &El[ne].nc)!=EOF)
    {
        if ((El[ne].na>=5) && (El[ne].nb>=5) &&
            (El[ne].nc>=5))
            El[ne].ar='A';
        else El[ne].ar='R';
        El[ne].md=(El[ne].na+El[ne].nb+El[ne].nc)/3.0;
        ne++;
        if (ne==na)
        {
            na+=NA;
            El=(StEl *)realloc(El,na*sizeof(StEl));
        }
    }
    fclose(fi);
    qsort(El,ne,sizeof(StEl),comp);
    for (i=0; i<ne; i++)
        printf("%-12s c%6.2lf\n", El[i].nm,El[i].ar,El[i].md);
    free(El);
    return 0;
}

```

3) Se citește dintr-un fișier text o valoare naturală n. Următoarele linii conțin n cuvinte, fiecare cuvânt având același număr de litere (cel mult 10). Să se afișeze cuvintele din fișier ordonate alfabetic.

Pentru a memora lista de cuvinte folosim următoarea strategie. În loc să alocăm pentru fiecare cuvânt (șir de caractere) citit o zonă nouă de memorie,

alocăm de la început o zonă în care să putem memoria toate cuvintele din listă. Această zonă va avea mărimea de  $(l+1) \cdot n$  octeți, unde  $l$  este lungimea fiecărui cuvînt (numărul de litere). De ce  $(l+1)$ ? pentru că trebuie să memorăm și caracterul terminator null.

Avantaje: memoria este utilizată mai eficient dacă se alocă de la început o zonă contiguă de dimensiune mai mare, și se reduce foarte mult fragmentarea memoriei.



```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
int comp(const void *A, const void *B)
{
    return strcmp((char *)A,(char *)B);
}
int main(int ac, char **av)
{
    char *C,s[11];
    int n,l,i;
    FILE *fi;
    if (ac!=2)
    {
        fputs("Un argument!\n",stderr);
        return 1;
    }
    fi=fopen(av[1],"rt");
    if (!fi)
    {
        perror("Eroare la deschidere");
        return 1;
    }
    fscanf(fi,"%d %s",n,s);
    l=strlen(s);
    C=(char *)malloc((l+1)*n);
    strcpy(C,s);
    for (i=1; i<n; i++)
        fscanf(fi,"%s",C+(l+1)*i);
    fclose(fi);
    qsort(C,n,l+1,comp);
    for (i=0; i<n; i++)
        printf("%s\n",C+(l+1)*i);
    free(C);
    return 0;
}
```

# CAPITOLUL 9

# Teste recapitulative

**N**ăștărirea și dezvoltarea cunoștințelor este un proces continuu. În acest capitol sunt prezentate câteva teste pentru verificare cunoștințelor acumulate prin parcursul tuturor noțiunilor și exemplelor din primele 8 capitole.



## TESTUL NUMĂRUL 1

1) Ce valori vor fi afişate în urma rulării următorului program?

```
#include <iostream.h>
```

```
int a[5], b[5], i;  
void main()  
{  
    for(i=0;i<5;i++)  
        if(i%2==0) a[i]=1;  
        else a[i]=i+2;  
    for(i=0;i<5;i++) b[i]=a[i]*2;  
    for(i=0;i<5;i++)  
        cout<<b[i]<<' ';
```

```
}
```

a) 1 6 1 10 1

b) 2 6 2 10 2

c) 2 2 6 2 2

d) 2 4 6 8 10

2) Se consideră următorul algoritm în pseudocod:

Citeşte n; max=0

Pentru i=0,9 executa

|      nr[i]=0;

■

Pentru i=1, n executa

|      Citeste j

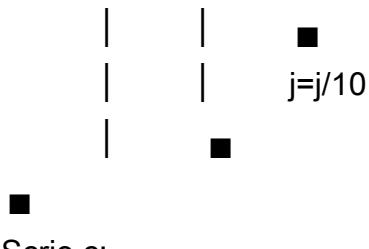
|      Cat timp j≠0 executa

|      |      x=j%10;

|      |      nr[x]=nr[x]+1;

|      |      daca max<nr[x] atunci

|      |      |      max=nr[x]; c=x;



Scrie c:

- a) Ce valoare va fi afișată pentru  $n=5$  și valorile 22, 235, 233, 6, 221?
  - b) Dați un exemplu pentru datele de intrare (nu toate nule), astfel încât să se afișeze valoarea 0.
  - c) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.
  - d) Realizați programul C++ corespunzător algoritmului dat.
- 3) Fie  $x$  un sir de numere reale. Se cere:
- a) Să se afișeze sirul ordonat crescător (metoda sortării prin selecție)
  - b) Să se eliminate numerele negative din vector. Să se afișeze sirul modificat.



## TESTUL NUMĂRUL 2

1) Ce valori vor fi afișate în urma rulării următorului program?

```
#include <iostream.h>
```

```
int a[5], i;  
void main()  
{  
    for(i=0;i<5;i++) a[i]=(i+1)*10;  
    for(i=1;i<5;i++) a[i]-=a[i-1];  
    for(i=0;i<5;i++) cout<<a[i]<<' ';
```

```
}
```

a) 10 20 30 40 40  
c) 10 10 10 10 10

b) 0 10 20 30 40  
d) 10 10 20 20 30

2) Se consideră următorul algoritm în pseudocod:

Citește n; max=1;

Pentru i=1, n executa

|      Citește a[i]

■

Scrie a[1]

Pentru i=2, n executa

|      Daca a[i]<a[i-1] atunci  
|      |      max=max+1  
|      |      scrie '\*', a[i];  
|      |      altfel scrie a[i];  
|      |      ■

■

Scrie max

- a) Ce valori vor fi afişate pentru  $n=7$  şi valorile **2,2, 34, 5, 6, 78, 8?**
- b) Dați un exemplu pentru datele de intrare (nu toate nule), astfel încât să nu se afișeze nici un caracter “\*”.
- c) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.
- d) Realizați un program în limbajul C++ corespunzător algoritmului dat.

3) Fie  $x$  un sir de numere reale. Se cere:

- a) Să se afișeze sirul ordonat descrescător (metoda bulelor);
- b) Să se insereze între oricare două elemente de pe poziții consecutive diferența lor în modul. Să se afișeze sirul modificat.



## TESTUL NUMĂRUL 3

1) Se consideră următoarea secvență repetitivă:

$i=0;$

$\text{while}(i+j \leq 10) \{i=i+1; j=j-2;\}$

Valoarea minimă posibilă pentru variabila  $j$  astfel încât instrucțiunea repetitivă de mai sus să nu se execute la infinit este:

a) 1

b) 5

c) 6

d) 17

e) 10

f) 2

2) Care dintre următoarele expresii are valoarea 1 dacă și numai dacă numărul real memorat în variabila  $x$  se află în intervalul  $(-2,2)$ ?

a)  $x^*x-4 \leq 0$

b)  $4-x^*x > 0$

c)  $(2 < x) \&\& (x < -2)$

d)  $(x-2)^*(x+2) > 0$

3) Variabilele  $x$  și  $y$  sunt de tip întreg,  $x$  memorând valoarea 8, iar  $y$  valoarea 6. Care dintre expresiile de mai jos are valoarea 0?

a)  $3*x-4*y == 0$

b)  $(x+y)/2 > x \% y + 1$

c)  $!(x/2+2 == y)$

d)  $x-y+3 != 0$

4) Se consideră următorul algoritm în pseudocod:

Citește  $x$  (număr natural)

$n=0$

cât timp  $x \neq 0$  execută

|     $y=x; c=0$

|    cât timp  $y > 0$  execută

|    |    dacă  $y \% 10 > c$  atunci

|    |    |     $c=y \% 10$

```

| | ■
| | y=[y/10]
| ■
| n=n*10+c
| citește x
■

```

Scrie n

- a) Scrieți valoarea ce se va afișa dacă se citesc, în această ordine, numerele **12, 7, 354, 9, 1630, 0**
  - b) Scrieți un set de date de intrare care să determine, în urma executării algoritmului, afișarea valorii **752**.
  - c) Scrieți programul C++ corespunzător algoritmului dat.
  - d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască fiecare structură **cât timp...execută** cu câte o structură repetitivă cu test final.
- 5) Să se realizeze algoritmul în pseudocod și programul în limbajul C++ corespunzător, pentru determinarea celui mai mare divizor comun a două numere naturale date, folosind pentru aceasta algoritmul lui Euclid.
  - 6) Se consideră un număr natural **n**. Să se afișeze cel mai mic multiplu par al numărului format din prima și ultima cifră a numărului dat. – programul în limbajul C++

*Exemplu:*

Pentru **n=1235** se va afișa **30**.



## **TESTUL NUMĂRUL 4**

1) Ce se va afisa la executarea urmatoarelor instructiuni?

p=1:

do {

$$p^*=2;$$

```
cout<<p;
```

```
} while(p%10!=6);
```



2) Variabila `x` este de tip real. Care dintre următoarele expresii C++ are valoarea 1 dacă și numai dacă numărul real memorat în variabila `x` nu aparține intervalului  $(2,9]$ ?

- a)  $(x > 2) \&\& (x \leq 9)$       b)  $(x \leq 2) \&\& (x > 9)$   
c)  $(x \leq 2) \mid\mid (x > 9)$       d)  $(x < 2) \mid\mid (x > 9)$

3) Care dintre următoarele expresii are valoarea 1 dacă și numai dacă variabilele  $x$  și  $y$  memorează două numere naturale pare consecutive?

- a)  $(x-y == 2) \&\& (y-x == 2)$       b)  $(x == 2) \&\& (y == 4)$   
c)  $x-y == 2$       d)  $((x-y == 2) || (y-x == 2)) \&\& (x \% 2 == 0)$

4) Se consideră următorul algoritm în pseudocod:

Citește n,k (numere naturale nenule)

nr=0; p=1;

cât timp  $n \neq 0$  și  $k \neq 0$  execută

| dacă  $n \% 2 \neq 0$  atunci

nr=nr+[n/10]\*10\*p

```

|   |     p=p*10
|   |     altfel
|   |     k=k-1
|   ■
|   n=[n/10]
■

```

Scrie nr

- Scriți valoarea care se afișează, în urma executării algoritmului, dacă se citește pentru **n** valoarea **23456** și pentru **k** valoarea **3**.
  - Scriți o pereche de numere naturale distințe, care pot fi citite atât pentru **n** cât și pentru **k**, astfel încât în urma executării algoritmului, să se afișeze valoarea **234**.
  - Scriți programul C++ corespunzător algoritmului dat.
  - Scriți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura **cât timp...execută** cu o structură repetitivă cu test final.
- Să se realizeze algoritmul în pseudocod și programul în limbajul C++ corespunzător, pentru determinarea celui mai mare divizor comun a două numere naturale date, folosind pentru aceasta algoritmul prin scăderi repetitive.
  - Se consideră un număr natural **n** ( $n > 100$ ). Să se afișeze suma primelor două cifre ale lui **n**. – programul în limbajul C++

*Exemplu:*

Pentru **n=12345** se va afișa **3**.



## TESTUL NUMĂRUL 5

1. Se consideră următorul algoritm în pseudocod:

Citește n {număr natural,  $n > 1$ }

ok=0;

cât timp  $n > 0$  execută

```
|   c=n%10;  
|   dacă c%2=1 atunci  
|       |           ok=1  
|       |           altfel  
|       |           ok=0  
|       □  
|   dacă ok=1 atunci  
|       |       scrie c,' '  
|       |       ok=1  
|       □  
|   n=[n/10]  
□
```

daca ok=0 atunci scrie "NU"

- a) Scrieți valoarea ce se afișează dacă  $n=9458$
- b) Scrieți cea mai mare valoare cu exact 3 cifre, care poate fi citită pentru  $n$  astfel încât să se afișeze, în această ordine, numerele 9 7.
- c) Scrieți programul C++ corespunzător algoritmului dat.
- d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura **cât timp...execută** cu o structură repetitivă de un alt tip.

2. Consideram urmatoarea secventa de program in care **v** este un vector format din **n** numere intregi, iar **n**, **i** si **aux** sunt trei variabile de tip intreg:

```
for(i=1;i<=n/2;i++)  
{      aux=v[i];  
      v[i]=v[n/2+n%2+i];  
      v[n/2+n%2+i]=aux;  
}
```

Daca **n=7** si **v=(10, 20, 30, 40, 50, 60, 70)** atunci dupa rularea secventei de mai sus vectorul **v** va avea urmatorul continut:

- a) **v=(40, 50, 60, 70, 10, 20, 30)**
- b) **v=(40, 50, 60, 10, 20, 30, 70)**
- c) **v=(50, 60, 70, 40, 10, 20, 30)**
- d) **v=(10, 20, 30, 70, 60, 50, 40)**

3. Se da o matrice patratica de ordinul **n**. Sa se formeze un vector cu **n** componente, in care componenta **i** a vectorului sa fie egala cu raportul dintre suma elementelor de pe linia **i** si suma elementelor de pe coloana **i**. Daca suma elementelor de pe coloana **i** este nula atunci componenta a **i**-a din vector va fi 0.

Exemplu: Daca matricea este

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

Se va obtine vectorul: 1.5, 0, 0.3, 1.1818



## TESTUL NUMĂRUL 6

1. Se consideră următorul algoritm în pseudocod:

Citește a,b {numere naturale}

c=0; d=0; p=1;

cât timp  $a+b+c>0$  execută

|        c=a%10+b%10+c; d=d+(c%10)\*p; p=p\*10;

|        a=[a/10]; b=[b/10]; c=[c/10];

□

Scrie d

- Scriți valoarea care se afișează dacă se citesc numerele **a=493** și **b=1836**.
- Scriți programul C++ corespunzător algoritmului dat.
- Scriți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura **cât timp...execută** cu o structură repetitivă de un alt tip.
- Scriți în pseudocod un algoritm echivalent cu cel dat, care să **NU** folosească structuri repetitive.

2. Considerând următoarele instrucțiuni identificati două echivalente:

- for( $i=0; i < n; i++$ ) if( $i \% 2 \neq 0$ )  $s = s + a[i]$ ;
- for( $i=1; i \leq n/2; i++$ )  $s = s + a[2^*i - 1]$ ;
- for( $i=n-1; i > 1; i--$ ) if( $i \% 2 \neq 0$ )  $s = s + a[i-1]$ ;
- for( $i=2; i \leq (n+1)/2; i++$ )  $s = s + a[i^*2-3]$ ;

3. Fie **a** o matrice cu **m** linii și **n** coloane cu numere reale și un vector **v** cu **n** componente reale. Sa se determine daca acest vector apare ca linie in matricea **a** si, in caz afirmativ, sa se afiseze numarul acestei linii. In caz contrar, sa se adauge vectorul ca ultima linia a matricei.

**Exemplu:**

Daca matricea A este:

1 3 6 2 9

7 4 0 3 8

5 2 1 47 6

Si vectorul este: 5, 2, 1, 47, 6 atunci se va afisa valoarea 3.

Daca matricea A este:

1 3 6 2 9

7 4 0 3 8

5 2 1 47 6

Si vectorul este: 2, 2, 1, 47, 6 atunci se va afisa valoarea matricea modificata:

1 3 6 2 9

7 4 0 3 8

5 2 1 47 6

2 2 1 47 6



## TESTUL NUMĂRUL 7

Cat timp  $n \geq 10$  executa

```
|      s=0  
|      Cat timp n≠0 executa  
|      |      s=s+n%10  
|      |      n=[n/10]  
|      n=s
```

Scrie n

- a) Ce se va afisa daca valoarea citita pentru n este 989736?
  - b) Stabiliți două numere diferite, de 5 cifre fiecare care, atribuite initial lui n, au ca efect afisarea valorii 1
  - c) Scrieti programul C++ corespunzator algoritmului dat.
  - d) Scrieti un algoritm echivalent cu algoritmul dat, dar care sa utilizeze un alt tip de structura repetitive
- 5) Se dau n numere naturale (n numar natural nenul dat). Se cere sa se calculeze si sa se afiseze suma numerelor care au suma cifrelor divizibila cu 7.

*Exemplu*

Pentru n=4 si numerele 15, 601, 815, 0 se va afisa 1416



## TESTUL NUMĂRUL 8

- 1) Care dintre urmatoarele expresii are valoarea 1 stiind ca variabilele a, b si c de tip intreg au valorile a=1, b=2, c=3, d=2?
- a)  $(a == c) \&\& (b || d)$       b)  $(b > c) || (c > 3)$   
c)  $((b == d) \&\& (a != 0)) || (b <= c)$       d)  $(b > c) \&\& a$
- 2) Care trebuie sa fie valoarea initiala a variabilei i de tip intreg pentru ca in urma executarii instructiunii urmatoare, pe ecran sa fie afisata sevenita de caractere \*\*\*\*\*?
- ```
while(i*5<1000)
{
    cout<<'*';
    i=i*2+10;
}
```
- a) 3      b) 11  
c) 13      d) 5
- 3) Ce se va afisa pe ecran in urma executarii urmatoarelor instructiuni, daca pentru variabila intreaga a se citesc, in ordine, numerele: 1234, 234, 52, 25, 5432, 819?
- ```
for(i=1;i<=6;i++)
{
    cin>>a;
    if (i%2==0) cout<<a/100%10;
    else cout<<a/10%10;
}
```
- a) 230241      b) 432221  
c) 220241      d) 325038

4) Fie urmatorul algoritm pseudocod:

Citeste a,b (numere naturale)

c=a%10

pentru i=1,b-1 executa

| c=c\*a

|\_ c=c%10

Scrie c

- a) Ce valoare afiseaza algoritmul pentru a=28 si b=10?
- b) Scrieti o pereche de valori de cate doua cifre pentru a si b astfel incat algoritmul sa afiseze valoarea 8.
- c) Scrieti programul C++ corespunzator algoritmului dat.
- d) Scrieti algoritmul pseudocod care sa fie echivalent cu cel dat si care sa contine un alt tip de structura repetitive.

5) Se dau n numere naturale (n numar natural nenul dat). Se cere sa se calculeze si sa se afiseze produsul numerelor care au un numar impar de cifre.

*Exemplu*

Pentru n=4 si numerele 1590, 6, 105, 0 se va afisa 0



## TESTUL NUMĂRUL 9

1) Fie sirul următor  $a=(103, 210, 2, 6777, 56, 209)$ . Să se precizeze ce afișează următoarea secvență de program:

```
int x=a[1];
for(int i=1;i<6;i++) a[i]=a[i+1];
a[6]=x;
for(int i=1;i<=6;i++) cout<<a[i]/10<<endl;
```

2) Ce valori vor fi afisate in urma rularii urmatorului program?

```
#include<iostream.h>
int a[6],i;
int main()
{
    for(i=1;i<=5;i++) a[i]=i-1;
    cout<<a[2]<<' ';
    cout<<a[a[2]]<<' '<<a[a[a[3]]]<<' ';
}
```



3) Se dă un sir de numere naturale. Se cere să se determine câte numere din sirul dat au suma cifrelor număr prim.

*Exemplu:*

Fie sirul  $x=(10, 24, 467, 23, 140)$ .

Se va afisa 3.

4) Fie  $x$  un sir cu elemente intregi. Se cere sa se construiasca alte doua siruri, unul care sa contina elementele din sirul dat care au un numar egal de cifre pare si impare, iar al doilea sir celelalte elemente.

*Exemplu:*

Fie sirul  $x=(100, 2116, 4670, 1423, 40)$ .

Se vor crea urmatoarele siruri:

$a=(2116, 1423)$  si  $b=(100, 4670, 40)$ .



## TESTUL NUMĂRUL 10

1) Fie sirul următor  $a=(103, 210, 2, 6777, 56, 209)$ . Să se precizeze ce afișează următoarea secvență de program:

```
int x=a[6];
for(int i=6;i>=2;i--) a[i]=a[i-1];
a[1]=x;
for(i=6;i>=1;i--) cout<<a[i]%10<<' ';
```

2) Ce valori vor fi afișate în urma rulării următorului program?

```
#include<iostream.h>
int a[5], b[5], i;
int main()
{
    for(i=0;i<5;i++)
        if(i%2==0) a[i]=1;
        else a[i]=i+2;
    for(i=0;i<5;i++) b[i]=a[i]*2;
    for(i=0;i<5;i++) cout<<b[i]<<' ';
}
```

- a) 1 6 1 10 1                            b) 2 6 2 10 2  
c) 2 2 6 2 2                            d) 2 4 6 8 10

3) Se dă un sir de numere naturale. Se cere să se afișeze numerele din sirul dat care sunt palindroame și au cel puțin 2 cifre.

*Exemplu:*

Fie sirul  $x=(10, 22, 4, 232, 1)$ .

Se vor afisa **22 232**.

4) Fie  $x$  un sir cu elemente întregi. Se cere să se construiască alte două siruri, unul care să conțină elementele din sirul dat care au cel puțin 4 divizori, iar unul care să conțină celelalte numere.

*Exemplu:*

Fie sirul  $x=(10, 2, 60, 13, 40)$ .

Se vor crea următoarele siruri:

$a=(10, 60, 40)$  și  $b=(2, 13)$ .



## TESTUL NUMĂRUL 11

- 1) Fiecare dintre variabilele întregi  $x$ ,  $y$  și  $t$  memorează câte un număr natural de cel mult 4 cifre. Știind că  $x < y$ , care dintre următoarele expresii are valoarea 1 dacă și numai dacă numărul memorat în variabila  $t$  nu aparține intervalului deschis  $(x,y)$ ?
- a)  $(t \leq x) \text{ || } (t \geq y)$       b)  $(t > x) \text{ || } (t < y)$   
c)  $(t \leq x) \text{ && } (t \geq y)$       d)  $(t > x) \text{ && } (t < y)$
- 2) Care dintre următoarele expresii are valoarea 1 dacă și numai dacă numărul natural nenul memorat în variabila  $x$  de tip **int** este divizibil cu 100?
- a)  $x \% 10 + x / 10 \% 10 == 0$       b)  $x / 100 == 0$   
c)  $x \% 10 + x / 10 == 0$       d)  $x \% 10 + x \% 10 / 10 == 0$
- 3) Variabilele  $x$  și  $y$  memorează câte un număr natural, cu exact 2 cifre. Care este valoarea expresiei  $x - y$  știind că fiecare dintre expresiile următoare are valoarea 1?
- $x / 10 == y \% 10$        $y / 10 == x \% 10$        $x / 10 == x \% 10 + 1$
- a) 0      b) 9  
c) 1      d) 11
- 4) Se consideră următorul algoritm în pseudocod:
- Citește  $n$  (număr natural)  
 $m=0$ ;  $p=1$ ;  
cat timp  $n>0$  executa  
|       $c=n \% 10$

```

|      daca c>0 atunci
|      |
|      |      c=c-1
|      |
|      |      ■
|      m=m+c*p; p=p*10;
|      n=[n/10]
|
|      ■

```

Scrie m

- a) Scrieți valoarea care se afișează dacă se citește numărul **n=5172**.
  - b) Scrieți programul C++ corespunzător algoritmului dat.
  - c) Scrieți în pseudocod, un algoritm echivalent cu cel dat, în care să se înlocuiască structura **cat timp...executa** cu o structură repetitivă de un alt tip.
  - d) Scrieți toate valorile distincte, fiecare având exact 4 cifre, care pot fi citite pentru variabila **n** astfel încât să se afișeze valoarea **2008** pentru fiecare dintre acestea.
- 5) Se dă un sir cu elemente numere întregi. Se cere să se eliminate din sirul dat numerele care prima cifră număr par. Să se afișeze sirul înainte și după modificare.

#### Exemplu

Pentru sirul 45 60 201 102 78 89 se vor afisa  
 45 60 201 102 78 89  
 102 78

- 6) Se dă un sir cu elemente reale. Se cere să se construiască un alt sir care conține elementele strict pozitive din sirul dat. Să se ordeneze descrescător sirul construit și să se afișeze.

#### Exemplu

Pentru sirul: 3, -90, 31, -7, -6, 0, 23 se va afisa 31 23 3

- 7) Din fișierul **matrice.in** se citesc de pe prima linie două numere naturale **m** și **n** care reprezintă numărul de linii, respectiv de coloane ale unei matrice de numere întregi. De pe următoarele linii se citesc elementele

matricei. Se cere să se calculeze și să se afișeze pentru fiecare linie a matricei numărul de elemente pare. Rezultatele obținute se vor afișa pe prima linie a fișierului **matrice.out**, despărțite de câte un spațiu.

**Exemplu**

**matrice.in**

3 4

34 2 9 67

1 3 19 7

2 6 20 2

**matrice.out**

2 0 4



## TESTUL NUMĂRUL 12

- 1) Care dintre variabilele întregi  $x$ ,  $y$  și  $z$  vor avea la finalul executării secvenței următoare de instrucțiuni, aceeași valoare ca înainte de executare?

$$x=y+z; z=x-z; y=z; z=x-y;$$

- a) numai  $x$  și  $z$       b) numai  $y$  și  $z$   
c) numai  $x$  și  $y$       d)  $x$ ,  $y$  și  $z$
- 2) Stabiliți care dintre următoarele expresii are valoarea 1 dacă și numai dacă numărul întreg  $x$  nu aparține intervalului  $(-10, -2) \cup [50, 100]$ .
- a)  $x \leq -10 \ || \ (x < 50 \ \&\& x \geq -2) \ || \ (x > 100)$   
b)  $x \leq -10 \ || \ (x \leq 50 \ \&\& x \geq -2) \ || \ (x \geq 100)$   
c)  $x < -10 \ || \ (x < 50 \ \&\& x > -2) \ || \ (x > 100)$   
d)  $x \leq -10 \ || \ (x \leq 50 \ || x \geq -2) \ || \ (x > 100)$
- 3) Variabilele  $n$ ,  $z$  și  $u$  sunt întregi, iar  $n$  memorează un număr natural cu cel puțin 2 cifre. Secvența care determină interschimbarea ultimelor două cifre din scrierea numărului memorat de  $n$  este:
- a)  $n=(n/100*10+n\%10)*10+n\%100/10;$   
b)  $u=n\%10; z=n/100\%10; n=n/100+u*10+z;$   
c)  $n=(n/100*10+n\%10)*10+n/100\%10;$   
d)  $u=n\%10; z=n/100\%10; n=n/100*100+z*10+u;$
- 4) Fie următorul algoritm în pseudocod:
- Citeste  $n$ ,  $d$  (numere naturale)  
 $b=0; v=0;$   
pentru  $i=1, n$  executa  
|     citeste  $x$  (numar natural nenul)

```

|   a=0;
|   aux=x;
|   cat timp x%d=0 executa
|       |   a=a+1
|       |   x=[x/d]
|
|       ┌─┐
|       daca a>b atunci
|           |   b=a;
|           |   v=aux;
|
|       ┌─┐
|
└─┘

```

Scrie v, b

- a) Scrieți ce se afișează dacă  $n=3$ ,  $d=2$ , iar valorile citite pentru  $x$  sunt, în ordine: 40, 19, 56.
  - b) Pentru  $n=3$  și  $d=2$  scrieți 3 valori distincte care pot fi citite în ordine pentru  $x$ , astfel încât valorile afișate să fie 0 și 0.
  - c) Scrieți programul C++ corespunzător algoritmului dat.
  - d) Scrieți în pseudocodul algoritm echivalent cu cel dat în care structura **cat timp...executa** să fie înlocuită cu o structură repetitive cu test final.
- 5) Se dă un sir cu elemente numere întregi. Se cere să se insereze între oricare două elemente situate pe poziții consecutive cel mai mare divizor comun al lor. Să se afișeze sirul înainte și după modificare.

#### Exemplu

Pentru sirul 4 6 20 1 8 9 se vor afisa

4 6 20 1 8 9

4 2 6 2 20 1 1 1 8 1 9

- 6) Se dă un sir cu elemente întregi. Se cere să se construiască un alt sir care conține elementele impare din sirul dat. Se cere să se ordoneze crescător sirul construit și să se afișeze.

### Exemplu

Pentru sirul: 3, -90, 31, -7, -6, 0, 23 se va afisa -7 3 23 31

- 7) Din fișierul **matrice.in** se citesc de pe prima linie două numere naturale **m** și **n** care reprezintă numărul de linii, respectiv de coloane ale unei matrice de numere întregi. De pe următoarele linii se citesc elementele matricei. Se cere să se calculeze și să se afișeze pentru fiecare coloană a matricei produsul elementelor impare. Rezultatele obținute se vor afișa pe prima linie a fișierului **matrice.out**, despărțite de către un spațiu.

### Exemplu

**matrice.in**

```
3 4  
34 2 9 67  
1 3 19 7  
2 6 20 2
```

**matrice.out**

```
1 3 171 469
```



## TESTUL NUMĂRUL 13

1) Se consideră graful neorientat dat prin următoarea matrice de adiacență.

0 1 1 0 0

1 0 1 0 0

1 1 0 0 0

0 0 0 0 1

0 0 0 1 0

Stabiliți care dintre următoarele afirmații este adevărată:

- a) Graful este conex;
- b) Prin adăugarea unei muchii graful devine conex;
- c) Graful nu prezintă ciclu;
- d) Prin eliminarea oricărei muchii graful nu prezintă ciclu.

2) Un arbore are nodurile numerotate de la 1 la 5. Care dintre următorii vectori poate fi vector de tați?

a) 4 4 1 0 1

b) 4 4 1 2 1

c) 2 3 0 4 3

d) 1 2 0 3 4

3) Fie  $G$  un graf orientat cu  $n$  noduri și  $m$  arce. Care este valoarea sumei gradelor exterioare ale tuturor nodurilor grafului?

a)  $2^*m$

b)  $n+m$

c)  $n$

d)  $m$



- a) Scrieți ce va afișa algoritmul dacă pentru **n** se citește valoarea **123611**.
  - b) Scrieți câte valori naturale distințe, formate din patru cifre fiecare, pot fi citite pentru variabila **n**, astfel încât, pentru fiecare dintre acestea, valoarea afișată de algoritm să fie divizibilă cu **10**.
  - c) Scrieți în pseudocod un algoritm echivalent cu cel dat care să utilizeze o singură structură repetitivă.
  - d) Scrieți programul C++ corespunzător algoritmului dat.
- 8) Se dă un graf neorientat prin numărul de vârfuri și matricea de adiacență. Se cere să se afișeze perechile de vârfuri între care există lanț în graful dat.
- 9) Se dă un arbore binar prin **numărul de noduri**, vectorii **S** și **D**. Se cere să se determine rădăcina, frunzele, matricea de adiacență, vectorii **T** și **DESC**.
- 10) Se dă un graf orientat prin numărul de noduri, numărul de arce și matricea de incidentă. Se cere să se construiască și să se afișeze matricea de adiacență. Pentru fiecare vârf al grafului, să se afișeze gradul extern și gradul intern.



## TESTUL NUMĂRUL 14

- 1) Se consideră graful neorientat cu 13 noduri și multimea muchiilor {[1,4], [2,5], [3,8], [4,7], [4,9], [4,11], [6,3], [6,10], [6,12], [8,6], [13,2]}. Identificați care sunt nodurile care formează componenta conexă cu număr maxim de noduri terminale.
- a) 3, 6, 8, 10, 12      b) 2, 5, 3, 6, 8, 10, 12  
c) 1, 4, 7, 9, 11      d) 2, 5
- 2) Pentru un arbore cu rădăcină având 9 noduri, care dintre următorii vectori ar putea fi vector de tați?
- a) (4,3,0,3,9,9,6,6,9)      b) (4,3,0,3,9,9,6,6,3)  
c) (4,3,2,3,9,9,6,6,3)      d) (4,3,2,3,9,9,6,6,0)
- 3) Fie graful orientat cu 5 noduri și arcele (1,2), (1,5), (2,5), (2,4), (3,2), (4,3), (4,5). Care este numărul minim de arce care trebuie adăugate grafului astfel încât să existe cel puțin un drum între oricare două vârfuri?
- a) 1      b) 0  
c) 3      d) 2
- 4) Un graf neorientat și conex are  $n$  noduri și  $n-1$  muchii. Care este numărul minim de muchii ce trebuie adăugate astfel încât să se obțină un ciclu?
- a)  $\frac{n^2 - 3n - 2}{2}$       b)  $\frac{n(n-1)}{2}$   
c) 0      d) 1

7) Fie următorul algoritm în pseudocod:

Citeste  $n$  (numar natural nenul)

Pentru  $i=1, n$  executa

```
| Citeste x (numar natural)
| nr=0
| cat timp x>0 executa
| | nr=nr*100+x%10
| | x=[x/100]
| |
| cat timp nr>0 executa
| | x=x*10+nr%10
| | nr=[nr/10]
| |
| Scrie x
```

- a) Scrieți valorile afișate dacă pentru  $n$  se citește valoarea 6, iar pentru  $x$  se citesc în ordine următoarele valori: 2008, 1965, 2727, 1861, 11021, 165.
- b) Știind că valoarea citită pentru  $n$  este 4, scrieți un set de valori distincte, numere naturale cu exact 3 cifre, care trebuie citite pentru variabila  $x$ , astfel încât setul de valori afișate în urma executării algoritmului să fie identic cu setul de valori citite pentru  $x$ .
- c) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura **pentru...execută** cu o structură repetitive cu test final.
- d) Scrieți programul C++ corespunzător algoritmului dat.
- 8) Se dă un graf neorientat prin numărul de noduri și matricea de adiacență. Se cere să se verifice dacă graful dat este sau nu un arbore.
- 9) Se dă un arbore binar prin **numărul de noduri**, vectorii  $T$  și **DESC**. Se cere să se determine rădăcina, matricea de adiacență, frunzele, vectorii  $S$  și  $D$ .
- 10) Se dă un graf orientat prin numărul de noduri și matricea de adiacență. Se cere să se construiască și să se afișeze matricea de incidentă. Pentru fiecare nod al grafului să se afișeze mulțimea succesorilor și mulțimea predecesorilor.



## TESTUL NUMĂRUL 15

- 1) Se dă un cuvânt format numai din litere mari. Se cere să se afișeze toate prefixele cuvântului, în două moduri:

a) de la prefixul cel mai mare la cel mai mic

MASCA

MASC

MAS

MA

M

b) de la prefixul cel mai mic la cel mai mare

M

MA

MAS

MASC

MASCA

- 2) Se dă un text. Se cere să se numere **vocalele, cifrele și semnele de punctuație** din text. Se știe că în textul dat pot fi următoarele **semne de punctuație**: virgulă, punct și virgulă, punct și semnul exclamării.

*Exemplu:*

“Lucrare „; de... ! control” – conține 6 vocale, 0 cifre și 7 semne de punctuație

- 3) Se dă un text și se cere ca fiecare literă mică urmată de o cifră să fie înlocuită de litera mare corespunzătoare, iar fiecare literă mare urmată

de spațiu să fie înlocuită de litera mică corespunzătoare. Să se afișeze textul înainte și după modificare.

*Exemplu:*

“Gj9ib0 fdD Y 78a” se va transforma astfel: “GJ9iB0 fdd y 78a”

- 4) Se dă un cuvânt format din litere mici. Se cere să se numere consoanele, iar fiecare vocală se fie înlocuită de caracterul ‘\*’. Să se afișeze numărul de consoane și textul modificat.

*Exemplu:*

“calculator” – 6 consoane; textul modificat va fi c\*Ic\*I\*t\*r



## TESTUL NUMĂRUL 16

- 1) Se dă un cuvânt format numai din litere mari. Se cere să se afișeze toate sufixele cuvântului, în două moduri:

a) de la sufixul cel mai mare la cel mai mic

MASCA

ASCA

SCA

CA

A

b) de la sufixul cel mai mic la cel mai mare

A

CA

SCA

ASCA

MASCA

- 2) Se dă un text. Se cere să se determine câte **litere mari**, câte **litere mici** și câte **cifre pare conține textul**.

*Exemplu:*

“Lucrare „; De... 891322 Control” – 3 litere mari, 13 litere mici și 3 cifre pare

- 3) Se dau două cuvinte formate din litere mici. Se cere să se construiască un sir din concatenarea celor două siruri date, în ordine lexicografică.

Sirul rezultat se va afișa mai întâi numai cu litere mari, iar apoi numai cu litere mici.

*Exemplu:*

Dacă se dau cuvintele “mouse” și “laptop”, se vor afișa:

LAPTOPMOUSE

laptopmouse

- 4) Se dă un cuvânt format din litere mici. Se cere să se numere vocalele, iar fiecare consoană se fie înlocuită de caracterul ‘&’. Să se afișeze numărul de vocale și textul modificat.

*Exemplu:*

“informatica” – 5 vocale; textul modificat va fi i&&o&&a&i&a



## TESTUL NUMĂRUL 17

1) Fie  $x$  un sir cu  $n$  valori naturale. Se cere sa se calculeze produsul numerelor din sir care sunt palindroame. Se vor folosi urmatoarele subprograme:

- Functie pentru citirea sirului;
- Functie care are ca parametru o variabila de tip intreg si care returneaza 1 daca valoarea variabilei este palindrom si 0 in caz contrar.

*Exemplu:* Pentru  $x=(2, 13, 44, 20)$  se va afisa **88**

2) Se da o matrice cu numere intregi. Se cere sa se afiseze cate numere divizibile cu 3 sunt pe fiecare linie a matricei. Se vor folosi urmatoarele subprograme:

- Functie pentru citirea matricei;
- Functie care are ca parametru un indice de linie si returneaza numarul de elemente divizibile cu 3 de pe linia respectiva.

*Exemplu:*

Pentru matricea

3 4 60 18

2 2 2 2

9 9 3 3

Se vor afisa valorile 2 0 4

3) Fie o matrice cu numere intregi. Se cere sa afiseze suma elementelor pare de pe diagonala principala a matricei si numarul de elemente strict negative din matrice. Se vor folosi urmatoarele subprograme:

- Functie pentru citirea matricei;
- Functie care returneaza suma elementelor pare de pe diagonala principala a matricei
- Functie care returneaza numarul de elemente strict negative din matrice.

*Exemplu:*

Pentru matricea

**3 -4 60 -18**

**2 2 2 2**

**9 9 3 3**

**1 1 -1 2**

Se vor afisa valorile **4 3**



## TESTUL NUMĂRUL 18

1) Fie  $x$  un sir cu  $n$  valori naturale. Se cere sa se calculeze suma elementelor din sir care sunt numere prime. Se vor folosi urmatoarele subprograme:

- Functie pentru citirea sirului;
- Functie care are ca parametru o variabila de tip intreg si care returneaza 1 daca parametrul este numar prim si 0 in caz contrar.

*Exemplu:*

Pentru  $x=(2, 13, 44, 20)$  se va afisa 15

2) Se da o matrice cu numere intregi. Se cere sa se afiseze cate numere impare sunt pe fiecare coloana a matricei. Se vor folosi urmatoarele subprograme:

- Functie pentru citirea matricei;
- Functie care are ca parametru un indice de coloana si returneaza numarul de elemente impare de pe coloana respectiva.

*Exemplu:*

Pentru matricea

3 4 60 1

2 2 2 25

9 9 3 3

Se vor afisa valorile 2 1 1 3

3) Fie o matrice cu numere intregi. Se cere sa afiseze numarul elementelor strict negative de pe diagonala secundara a matricei si produsul elementelor din matrice care sunt strict pozitive si au exact o cifra. Se vor folosi urmatoarele subprograme:

- Functie pentru citirea matricei;
- Functie care returneaza numarul elementelor strict negative de pe diagonala secundara a matricei
- Functie care returneaza produsul elementelor din matrice care sunt strict pozitive si au exact o cifra.

*Exemplu:*

Pentru matricea

**3 -4 60 -18  
2 2 2 2  
9 9 3 3  
-1 1 -1 2**

Se vor afisa valorile **2 69984**



## **TESTUL NUMĂRUL 19**

- 1) Variabila **s** memorează un sir de caractere. Care dintre următoarele expresii are valoare **nenulă** dacă și numai dacă lungimea sirului este strict mai mică decât 10?

a) `strlen(s)<10`      b) `strlen(s,10)<0`  
c) `leng(s)<10`      d) `s-'0'<10`

2) Știind că în urma executării secvenței de program următoare s-a afișat succesiunea de caractere **EXAMEN**, care este sirul de caractere memorat de variabila **s**?

```
x=strlen(s);  
for(i=0;i<x/2;i++)  
cout<<s[i]<<s[x-i-1];
```

a) ENXAME      b) EAENMX  
c) NEEEXMA      d) NEMAXE

3) Care este valoarea variabilei **s** de tip sir de caractere după executarea instrucțiunii de mai jos?

```
strcpy(s,strcat(strchr("bacalaureat", 'b')+strlen("2012"), "12"));
```

a) BAC2012      b) laureat12  
c) bac201212      d) aur2012

4) Variabila **s** reține sirul de caractere **"informatica"**. Ce se afișează la executarea instrucțiunii de mai jos?

```
cout<<strrchr(s, 'i');
```

- 5) Ce valoare se va afișa pe ecran în urma executării secvenței de program următoare?

```
strcpy(a,"culegere");
for(i=0;i<strlen(a);i++)
a[i]=a[i]-1;
cout<<a;
```

- 6) Fie **a** un sir de caractere care reține textul “Primavara”. Ce se va afișa în urma executării următoarei secvențe de instrucțiuni?

```
for (int i=1;i<=3;i++)
strcpy(a+1,a+2);
cout<<a;
a) Pavara
b) ara
c) rim
d) Para
```

- 7) Ce se va afișa la finalul executării următoarei secvențe de instrucțiuni?

Variabilele **x** și **p** sunt de tip sir de caractere.

```
strcpy(x,"albacazapada");
x[0]=x[0]-32;
strcpy(p, strchr(x,'a'));
cout<<x[0]<<p[0]<<x[strlen(x)-1];
a) aaa
b) AaA
c) Aaa
d) AAA
```



# **TESTUL NUMĂRUL 20**

- 1) Ce se afișează pe ecran în urma executării secvenței următoare știind că variabila `i` este de tip `char`?

```
for(i='a';i<='z';i++)  
    if(strchr("calculator"),i) cout<<i;
```

- 2) În secvența următoare, fiecare dintre variabilele **x** și **s** sunt de tip sir de caractere, iar **i** este de tip întreg. Dacă variabilele **x** și **s** memorează inițial sirul **absolvent**, ce se va memora în variabila **x** în urma executării secvenței următoare?

```
for(i=0;i<strlen(s);i++)  
if(strcmp(x,s+i)<0) strcpy(x,s+i);  
  
a) nt  
c) solvent
```

- 3) Știind că variabila `i` este de tip întreg și variabila `s` reține un sir de caractere, ce se va afisa la executarea secvenței următoare?

```
strcpy(s,"ClasA10BLucrArEdecONTRollaINFORMATICA");
```

```
for(i=0;i<strlen(s);i++)  
if(s[i]<'a' || s[i]>'z') cout<<s[i];  
  
a) 10  
b) lasucrrdecolla  
c) CA10BLAEONTRINFORMATICA d) CABLAEONTRINFORMATICA
```

- 4) Ce memorează variabila **s**, de tip sir de caractere, după executarea instrucțiunii de mai jos?

```
strncpy(s, "informatica", strlen("2009")); s[strlen("2009")]='\0';
strcat(s,"BAC");
a) info b) infoBAC
c) BACinfo d) informaticaBAC
```

5) Ce se afișează în urma executării secvenței următoare, în care variabila **c** memorează un sir cu cel mult 20 de caractere, iar variabila **i** este de tip întreg?

```
char c[]={tamara};
for(i=0;i<3;i++) c[i]=c[i+1];
cout<<c;
```

6) Ce se va afișa în urma executării următoarei secvențe de instrucțiuni?

```
strcpy(x,"Mama"); strcpy(y,"Macara");
if(strcmp(x,y)>0) cout<<x;
else if(strcmp(x,y)==0) cout<<"Incorect";
else cout<<y;
```

a) Macara b) Mama  
c) MamaIncorect d) Incorect

7) Fie secvența de instrucțiuni următoare:

```
for(i=0;i<strlen(a);i++) if(a[i]>='A'&&a[i]<='Z') a[i]+=32;
```

Ştiind că **a** este un sir de caractere și **i** o variabilă de tip întreg identificați prelucrarea realizată asupra caracterelor sale.

a) Transformarea literelor mici în literele mari corespunzătoare;  
b) Inserarea sirului de caractere 32 după fiecare caracter de tip majusculă;  
c) Transformarea literelor mari în literele mici corespunzătoare;  
d) Ordonarea alfabetică a majuscuilelor în cadrul sirului.



# TESTUL NUMĂRUL 21

- a) Ce se va afișa dacă valoarea citită pentru  $n$  este 989736?
- b) Stabiliți două numere diferite, de 5 cifre fiecare care, atribuite inițial lui  $n$ , au ca efect afișarea valorii 1
- c) Scrieți programul C++ corespunzător algoritmului dat.
- d) Scrieți un algoritm echivalent cu algoritmul dat, dar care să utilizeze un alt tip de structură repetitivă
- 4) Se consideră un tablou bidimensional cu  $n$  linii și  $m$  coloane ( $1 \leq n, m \leq 24$ ) ce memorează numere întregi cu cel mult două cifre fiecare. Scrieți un program care citește datele de intrare din fișierul "matrice.in" și care inversează ordinea elementelor în cadrul fiecărei coloane, ca în exemplul următor. Programul va afișa în fișierul de ieșire "matrice.out" matricea obținută după inversare, fiecare linie a matricei pe câte un rând de fișier, iar elementele aceleiași linii separate prin câte un spațiu.

*Exemplu:*

1 7 3	devine	3 4 5
4 5 6		7 8 9
7 8 9		4 5 6
3 4 5		1 7 3



## TESTUL NUMĂRUL 22

- 1) Variabilele **a**, **b** și **y** sunt reale, iar **a≤b**. Care dintre expresiile următoare are valoarea 1 dacă și numai dacă valoarea variabilei **z** nu aparține intervalului închis determinat de valorile variabilelor **a** și **b**?
- a)  $(z > a \text{ || } z > b)$       b)  $(z < a) \text{ || } (z > b)$   
c)  $z < a \text{ && } z > b$       d)  $z \geq a \text{ && } z \leq b$
- 2) Care dintre următoarele instrucțiuni atribuie variabilei întregi **t** valoarea -1 dacă și numai dacă variabilele întregi **a** și **b** sunt nenule și au semne diferite?
- a) if ((a>0)|| (b<0)) t= -1;      b) if ((a>0)&&(b<0)) t= -1;  
c) if(a\*b<0) t= -1;      d) if (a\*b>0) t= -1;
- 3) Fie următorul algoritm pseudocod:
- Citeste **a,b** (numere naturale)  
**c=a%10**  
pentru **i=1,b-1** executa  
|      **c=c\*a**  
|\_     **c=c%10**  
Scrie **c**
- a) Ce valoare afișează algoritmul pentru **a=28** și **b=10**?  
b) Scrieți o pereche de valori de câte două cifre pentru **a** și **b** astfel încât algoritmul să afișeze valoarea 8.  
c) Scrieți programul C++ corespunzător algoritmului dat.  
d) Scrieți algoritmul pseudocod care să fie echivalent cu cel dat și care să conțină un alt tip de structură repetitivă.

- 4) Scrieți un program care citește din fișierul “**date.in**” un număr natural  $n$  ( $n \leq 10$ ) apoi construiește în memorie o matrice cu  $2*n$  linii și  $2*n$  coloane, numerotate de la 1 la  $2*n$ , astfel încât parcurgând doar liniile impare ale matricei de sus în jos și fiecare linie impară de la stânga la dreapta se obțin în ordine crescătoare toate numerele cuprinse în intervalul  $[1, 4*n^2]$ , iar parcurgând doar liniile pare ale matricei de sus în jos și fiecare linie pară de la dreapta la stânga se obțin în ordine strict crescătoare toate numerele pare cuprinse în intervalul  $[1, 4*n^2]$ , ca în exemplu.

Programul afișează în fișierul “**date.out**” matricea obținută, câte o linie a matricei pe câte o linie a fișierului, elementele fiecărei linii fiind separate prin câte un spațiu.

*Exemplu:*

Pentru  $n=2$  se obține matricea următoare:

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



## TESTUL NUMĂRUL 23

- 1) Care este valoarea pe care trebuie să o aibă inițial variabila întreagă  $x$  pentru ca în urma executării secvenței următoare, să se afișeze sirul:  
HHHHHH?

while ( $x!=3$ ) { $x=x-1$ ; cout<<"HH";}

- a) 0
- b) 4
- c) 6
- d) 5

- 2) Variabilele  $n$ ,  $z$  și  $u$  sunt întregi, iar  $n$  memorează un număr natural cu cel puțin două cifre. Secvența care determină interschimbarea ultimelor două cifre din scrierea numărului memorat de  $n$  este:
- a)  $n=(n/100*10+n\%10)*10+n\%100/10;$
  - b)  $u=n\%10; z=n/100\%10; n=n/100+u*10+z;$
  - c)  $n=(n/100*10+n\%10)*10+n/100\%10;$
  - d)  $u=n\%10; z=n/100\%10; n=n/100*100+z*10+u;$

- 3) Se consideră următorul algoritm pseudocod:

Citeste  $n$  (număr natural cu cel mult 9 cifre)

Cat timp  $n \geq 10$  executa

```
|      s=0
|      Cat timp n≠0 executa
|      |      s=s+n%10
|      |      n=[n/10]
|      n=s
```

Scrie  $n$

- a) Ce se va afișa dacă valoarea citită pentru  $n$  este 989736?

- b) Stabiliți două numere diferite, de 5 cifre fiecare care, atribuite inițial lui  $n$ , au ca efect afișarea valorii 1
- c) Scrieți programul C++ corespunzător algoritmului dat.
- d) Scrieți un algoritm echivalent cu algoritmul dat, dar care să utilizeze un alt tip de structură repetitivă
- 4) Se consideră un tablou bidimensional cu  $n$  linii și  $m$  coloane ( $1 \leq n, m \leq 24$ ) ce memorează numere întregi cu cel mult două cifre fiecare. Scrieți un program care citește datele de intrare din fișierul "matrice.in" și care inversează ordinea elementelor în cadrul fiecărei coloane, ca în exemplul următor. Programul va afișa în fișierul de ieșire "matrice.out" matricea obținută după inversare, fiecare linie a matricei pe câte un rând de fișier, iar elementele aceleiași linii separate prin câte un spațiu.

*Exemplu:*

1 7 3	devine	3 4 5
4 5 6		7 8 9
7 8 9		4 5 6
3 4 5		1 7 3



## TESTUL NUMĂRUL 24

- 1) Variabilele **a**, **b** și **y** sunt reale, iar **a≤b**. Care dintre expresiile următoare are valoarea 1 dacă și numai dacă valoarea variabilei **z** nu aparține intervalului închis determinat de valorile variabilelor **a** și **b**?
- a)  $(z > a \text{ || } z > b)$       b)  $(z < a) \text{ || } (z > b)$   
c)  $z < a \text{ && } z > b$       d)  $z >= a \text{ && } z <= b$
- 2) Care dintre următoarele instrucțiuni atribuie variabilei întregi **t** valoarea -1 dacă și numai dacă variabilele întregi **a** și **b** sunt nenule și au semne diferite?
- a) if ((a>0)|| (b<0)) t= -1;      b) if ((a>0)&&(b<0)) t= -1;  
c) if(a\*b<0) t= -1;      d) if (a\*b>0) t= -1;
- 3) Fie următorul algoritm pseudocod:
- Citeste a,b (numere naturale)  
 $c=a \% 10$   
pentru i=1,b-1 executa  
|       $c=c * a$   
|\_      $c=c \% 10$   
Scrie c
- a) Ce valoare afișează algoritmul pentru **a=28** și **b=10**?  
b) Scrieți o pereche de valori de câte două cifre pentru **a** și **b** astfel încât algoritmul să afișeze valoarea 8.  
c) Scrieți programul C++ corespunzător algoritmului dat.  
d) Scrieți algoritmul pseudocod care să fie echivalent cu cel dat și care să conțină un alt tip de structură repetitivă.

4) Scrieți un program care citește din fișierul “**date.in**” un număr natural  $n$  ( $n \leq 10$ ) apoi construiește în memorie o matrice cu  $2*n$  linii și  $2*n$  coloane, numerotate de la 1 la  $2*n$ , astfel încât parcurgând doar liniile impare ale matricei de sus în jos și fiecare linie impară de la stânga la dreapta se obțin în ordine crescătoare toate numerele cuprinse în intervalul  $[1, 4*n^2]$ , iar parcurgând doar liniile pare ale matricei de sus în jos și fiecare linie pară de la dreapta la stânga se obțin în ordine strict crescătoare toate numerele pare cuprinse în intervalul  $[1, 4*n^2]$ , ca în exemplu.

Programul afișează în fișierul “**date.out**” matricea obținută, câte o linie a matricei pe câte o linie a fișierului, elementele fiecărei linii fiind separate prin câte un spațiu.

*Exemplu:*

Pentru  $n=2$  se obține matricea următoare:

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

## Bibliografie

- [1] Brian Kernighan, Dennis Ritchie - *The C Programming Language*; Editura Prentice Hall 1978, 1985
- [2] Bjarne Stroustrup - *The C++ Language (second edition)*; Editura Addison Wesley, 1993
- [3] Margaretz Ellis, Bjarne Stroustrup - *The Annotated C++ Reference Manual*; Editura Addison Wesley, 1991
- [4] *Borland C++. Version 4.5 - Programmers Guide*, Borland International, 1995
- [5] Kacso Adrian, Kacso Daniela - *Turbo C - Tehnici de programare*; Editura MicroInformatica, Cluj, 1994
- [6] Ionut Muslea - *Initiere în C++. Programare orientata pe obiecte*; Editura MicroInformatica, Cluj, 1992
- [7] Ionut Muslea - *C++ pentru avansati*; Editura MicroInformatica, Cluj, 1994
- [8] Octavian Catrina, Iulian Cojocaru – *Turbo C++*; Editura Teora, Bucuresti, 1993
- [9] Cristian Giumale, Eugenia Kalisz, Alexandru Plunoiu, Valentin Cristea - *Introducere în limbajul C*; Editura Teora, Bucuresti, 1992
- [10] Clara Ionescu, Ioan Zsako - *Structuri arborescente cu aplicatiile lor*; Editura Tehnică, Bucuresti, 1990
- [11] Dan Somnea si Doru Turturea - *Initiere în C++: Programarea orientata pe obiecte*; Editura Tehnica, Bucuresti, 1990
- [12] Nabajyoti Barakakati - *Borland C++ 4; Ghidul programatorului*; (traducere) Editura Teora, Bucuresti, 1996
- [13] Herbert Schilldt - *C++ manual complet*; (traducere), Editura Teora, Bucuresti, 1997
- [14] Herbert Schilldt - *C manual complet*; (traducere), Editura Teora, Bucuresti, 1998
- [15] Mickey Williams - *Bazele Visual C++ 4*; (traducere), Editura Teora, Bucuresti, 1997

- [16] Maria Codrina, Dana Lica, Doru Popescu Anastasiu, Radu Boriga, Adrian Runcceanu, Mihaela Runcceanu, Anca Voicu, Mariana Ciobanu - *Bacalaureat la informatică. Noțiuni recapitulative. Teze rezolvate și probleme propuse*; Editura L&S Infomat, București, 2001, ISBN 973-99376-4-0
- [17] Adrian Runcceanu - *Tehnici de programare - Îndrumar de laborator*, Editura Academica Brâncuși, Târgu-Jiu, 2002, ISBN 973-85805-7-9
- [18] Adrian Runcceanu - *Programarea și utilizarea calculatoarelor*, Editura Academica Brâncuși, Târgu-Jiu, 2003, ISBN 973-8436-44-3
- [19] Adrian Runcceanu - *Metode și tehnici de programare. Limbajul C++ - Îndrumar de laborator*, Editura Academica Brâncuși, Târgu-Jiu, 2003, ISBN 973-8436-37-9
- [20] Adrian Runcceanu, Mihaela Runcceanu - *Atestat la informatică*, Editura Axioma Teomsnic, Târgu-Jiu, 2004, ISBN 973-86389-7-X
- [21] Adrian Runcceanu - *Internet și Intranet. Teorie și aplicații*, Editura Academica Brâncuși, Târgu-Jiu, 2005, ISBN 973-7637-21-6
- [22] Adrian Runcceanu - *Programare orientată pe obiecte*, Editura Academica Brâncuși, Târgu-Jiu, 2007, ISBN (13) 978-7637-89-5
- [23] Marian Popescu, Adrian Runcceanu - *Baze de date – Visual Foxpro 6.0 – Îndrumar de laborator*, Editura Academica Brâncuși, Târgu-Jiu, 2007, ISBN 978-973-144-008-8
- [24] Adrian Runcceanu - *Programare orientată pe obiecte-limbajul C++. Laborator*, Editura Academica Brâncuși, Târgu-Jiu, 2008, ISBN 978-973-144-109-2
- [25] Adrian Runcceanu, Mihaela Runcceanu - *Tehnologii și aplicații web – Îndrumar de laborator*, Editura Academica Brâncuși, Târgu-Jiu, 2009, ISBN 978-973-144-302-7
- [26] Adrian Runcceanu, Mihaela Runcceanu - *Baze de date – o abordare Visual Foxpro*, Editura Academica Brâncuși, Târgu-Jiu, 2009, ISBN 978-973-144-235-8
- [27] Adrian Runcceanu - *Grafică asistată de calculator. Teorie și aplicații*, Editura Academica Brâncuși, Târgu-Jiu, 2009, ISBN 978-973-144-301-0