

TIPURI DE DATE UTILIZATE ÎN PRELUCRĂRI

1

STUDIU DE CAZ

Ionuț este solicitat de către profesorul diriginte să construiască o agendă într-o aplicație instalată pe calculatoarele laboratorului de Informatică. În această agendă trebuie să păstreze și să actualizeze datele personale și situația școlară ale fiecărui elev din clasă. Prin consultarea acestei agende, profesorul diriginte trebuie să obțină informațiile necesare la un moment dat.

Rezolvare

Pasul 1. Ionuț separă datele personale de situația școlară, pentru fiecare elev. El creează astfel două subagende. Consultarea celor două subagende se va face prin numărul de identificare primit de fiecare elev.

Subagenda *Date personale* conține următoarele informații:

Nr_iden.	Nume	Prenume	Adresa	E-mail	Telefon fix	Telefon mobil	Data nașterii
----------	------	---------	--------	--------	-------------	---------------	---------------

Din această agendă se pot afla:

- telefonul oricărui elev,
- elevii născuți în luna curentă,
- elevii care împlinesc o anumită vârstă.

Subagenda *Situație școlară* conține următoarele informații:

Nr_iden.	Medie sem I	Nr. Corig. Sem I	Abs. Nem. Sem I	Nota Purtare Sem I	Medie sem. II	Nr. Corig. Sem II	Abs. Nem. Sem II	Nota Purtare Sem II	Medie gen.	Promovat (da/nu)
----------	-------------	------------------	-----------------	--------------------	---------------	-------------------	------------------	---------------------	------------	------------------

Din această subagendă se pot afla:

- media generală a unui elev,
- dacă un elev este promovat sau nu,
- numărul total de absențe ale fiecărui elev.

Pasul 2. După ce a stabilit structura fiecărei agende, Ionuț întocmește o listă cu prelucrările pe care urmează să le facă:

- introducerea datelor personale,
- completarea rezultatelor școlare la sfârșitul fiecărui semestru,
- determinarea situației școlare la sfârșit de an școlar,
- afișarea rezultatelor în ordinea descrescătoare a mediilor.

Pasul 3. Ionuț observă că informațiile păstrate în cele două subagende diferă din punct de vedere al tipurilor de valori: șiruri de caractere (numele, prenumele), numere cu zecimale (mediile generale), date calendaristice (data nașterii), numere naturale (numărul de identificare).

Ionuț vrea să știe mai multe despre tipul datelor și caracteristicile acestora.

1. Etapele rezolvării problemelor



Orice problemă care va fi rezolvată cu calculatorul necesită o analiză atentă atât a cerințelor, a datelor care sunt prelucrate, cât și a condițiilor care trebuie respectate în rezolvarea problemei.

Înainte de a trece la rezolvarea problemei într-o anumită aplicație de calculator, este necesar să se parcurgă următoarele etape:

Etapa 1: Analiza problemei și identificarea datelor

În această etapă se stabilesc datele cunoscute (*date de intrare*) și datele solicitate prin cerințele problemei (*date de ieșire*).

Datele de intrare, datele de ieșire cât și alte date utilizate în prelucrările curente vor primi un *nume* (denumit **identificator**), care să le reprezinte în mod unic.

Etapa 2: Soluția problemei – algoritmul de rezolvare

Soluția problemei rezultă din raționamentul prin care datele de intrare sunt prelucrate pentru obținerea datelor de ieșire.

Acest raționament se numește *algoritm*, dacă respectă următoarele proprietăți:

- să fie cât mai simplu (să cuprindă operații elementare);
- să fie cât mai clar (să fie exprimat prin cuvinte cheie/instrucțiuni cu semnificație cunoscută);
- să furnizeze datele de ieșire corecte după un număr finit de pași (corectitudine și finitudine);
- să permită rezolvarea unor grupe de probleme cu caracteristici asemănătoare (generalitate).

Etapa 3: Reprezentarea algoritmului

Pentru testarea și codificarea raționamentului, acesta trebuie reprezentat într-o formă cât mai simplă și ușor de interpretat. Reprezentarea algoritmilor se poate face în schema logică (reprezentare grafică), în *pseudocod* sau în limbaj de programare.

Etapa 4: Testarea algoritmului

Se verifică pentru date de intrare reprezentative dacă algoritmul generează datele de ieșire corespunzătoare. Dacă pentru un anumit set de date de intrare algoritmul nu este corect, atunci se trece la refacerea algoritmului; se reiau etapele 2, 3, 4, până când rezultatele algoritmului sunt corecte.

Etapa 5: Implementarea algoritmului în aplicația dorită, apelând la facilitățile oferite de această aplicație (operații, tipuri de date, instrucțiuni etc.).

Etapa 6: Verificarea rezultatelor

Dacă sunt erori, atunci se controlează setul de operații, setul de tipuri de date și operatorii corespunzători, setul de instrucțiuni cu care este înzestrată aplicația curentă și se reface algoritmul sau doar implementarea acestuia, după caz.

2. Tipuri de date

În etapa de analiză a problemei se stabilesc:

- **datele de intrare** – date cunoscute din enunțul problemei (nume, prenume, note);
- **datele de ieșire** – date pe care trebuie să le furnizeze algoritmul, descoperite din cerințele problemei (medie generală, situație);
- **datele temporare** (auxiliare) date necesare pentru a obține datele de ieșire pe baza datelor de intrare (numărul de elevi corigenți pe primul semestru, valori temporare).

Pe parcursul algoritmului, unele date își pot modifica sau nu valoarea. Din acest punct de vedere, datele pot fi:

- **constante** – date care nu-și modifică valoarea pentru oricare set al datelor de intrare (anul curent, pentru toate prelucrările ce au loc într-un an calendaristic; unități de măsură; constante fizice sau matematice);
- **variabile** – date care își modifică valoarea (numărul de absențe, adresa, adresa de e-mail, numărul de telefon).

O variabilă sau o constantă poate fi personalizată printr-un nume sau **identificator**, astfel încât să poată fi apelată de mai multe ori în algoritm. Identificatorul este o succesiune de litere și cifre (primul caracter trebuie să fie o literă). Singurul separator acceptat este caracterul ' _ '.

După personalizarea datelor, este necesar să se stabilească mulțimea valorilor pe care le poate lua o dată, respectiv operațiile permise cu acestea. Se spune că se stabilește **tipul** datei respective. O dată poate reține valori corespunzătoare tipului asociat.

În funcție de tipul lor, datele pot fi clasificate astfel:

- *numerice* (numere naturale, întregi, reale);
- *caractere* (litere, cifre, semne de punctuație, simboluri speciale);
- *șiruri de caractere*;
- *logice* cu semnificația de adevărat sau fals (promovat sau nu, are 18 ani sau nu);
- *date calendaristice* (data nașterii), momente de timp (ora de începere a cursurilor);
- *speciale (generale)*: imagini, muzică, text (în subagenda date personale se pot atașa: fotografia fiecărui elev, melodia preferată, un text în care să fie trecute hobby-urile, realizările deosebite ale elevului).

În Tabelul 1 sunt prezentate tipurile de date și operațiile specifice acestora.

Tipuri de de date	Operații specifice
Numerice – naturale – întregi – reale	– operații aritmetice (*, /, +, –) – comparații – prelucrări algoritmice (determinarea parității, cel mai mare divizor comun, verificarea unor proprietăți)
Caracter/șiruri de caractere	– comparații – prelucrări specifice: conversii litere mari/mici, căutare, inserare, eliminare
Logice – adevărat (A) – fals (F)	– operații logice (NOT, AND, OR)
Data calendaristică (zi, lună, an) și timp (h, m, s)	– comparații – prelucrări specifice: determinări de intervale, momente de timp
Generale: – imagini – sunete	– procesări specifice pentru obținerea efectelor de culoare, rezoluție, amplitudine, frecvență ș.a.

Tabelul 1. Tipuri de date și operații specifice

Datele pot fi transformate/prelucrate prin operații simple compuse în **expresii**. O expresie reprezintă o succesiune de *operanzi* și *operatori*. Un *operand* poate fi o constantă sau o variabilă sau o altă expresie delimitată prin paranteze rotunde. *Operatorii* care pot fi utilizați într-o expresie depind de tipul operanzilor (întregi, reali, logici etc.).

3. Corectitudinea datelor

În cazul agendei personale pe care trebuie să o construiască Ionuț, toate informațiile despre un elev reprezintă o *instanță*, care definește unic elevul.

Datele memorate în agenda telefonică trebuie să fie corecte, să păstreze semnificația reală.

Din enunțul problemei se pot identifica condiții (restricții) pe care trebuie să le îndeplinească datele de intrare. *Datele de intrare sunt corecte sau valide* dacă respectă condițiile impuse de enunțul problemei numite *condiții de validare*.

Exemple:

- nota unui elev trebuie să fie un număr natural din intervalul [1, 10];
- anul nașterii unei persoane nu poate fi mai mare decât anul curent.

TEME

1. Cabinet medical.

Într-un cabinet medical se rețin date despre pacienții care sunt consultați. Fiecare pacient primește un număr de înregistrare. În registrul de consultații se trec următoarele informații: numele și prenumele, adresa, sexul, data și ora

consultației, diagnosticul, temperatura, tensiunea arterială, dacă se eliberează o rețetă atunci se trece numărul acesteia, dacă se eliberează o trimitere către un medic specialist, se înregistrează specialitatea.

La sfârșitul zilei, trebuie făcută următoarea situație:

- numărul pacienților consultați;
- numele pacienților care au primit rețete;
- numărul pacienților care au primit trimitere către un medic specialist.

Cerință:

După analiza problemei, completați un tabel după modelul de mai jos:

Date de intrare	Date de ieșire	Tipuri de date	Evenimente	Condiții de validare
-----------------	----------------	----------------	------------	----------------------

2. Bilanțul vânzărilor.

La un magazin de produse alimentare, directorul analizează activitatea zilnică a vânzărilor, a stocului de produse existent, data expirării produselor. Fiecare produs este caracterizat prin: cod, nume produs, preț unitar, cantitate vândută, TVA (16%, 19%, 23%), valoarea = cantitate*preț unitar*(1+TVA/100), cantitate existentă în stoc (la fiecare se scade din stocul existent cantitatea vândută din acel produs), data expirării. La sfârșitul fiecărei zile, directorul solicită:

- a) suma totală încasată;
- b) lista produselor care nu mai sunt în stoc;
- c) lista produselor expirate;
- d) produsele cele mai solicitate.

Cerință:

După analiza problemei, completați tabelul de mai jos.

Date de intrare	Date de ieșire	Tipuri de date	Evenimente	Condiții de validare
-----------------	----------------	----------------	------------	----------------------

3. Formulați câte o problemă, după modelul temelor 1 și 2, care să pună în evidență datele și prelucrările specifice următoarelor situații:

- a) activitatea de împrumut a cărților într-o bibliotecă;
- b) activitatea unei case de schimb valutar;
- c) rezultatele obținute de elevii participanți la un concurs sportiv;
- d) rezultatele obținute de elevii unui liceu în urma susținerii examenului de bacalaureat.

4. Operații elementare asupra datelor

Pentru prelucrarea algoritmică a datelor, se folosesc operații elementare precum:

- operații de intrare/ieșire;
- operații de atribuire;
- operații aritmetice, relaționale sau condiționale.

4.1. Operații de intrare – ieșire

Prin *operația de intrare* (sau operație de citire), valorile corespunzătoare datelor de intrare sunt preluate de la un dispozitiv de intrare (tastatură, dischetă, CD etc.) și sunt trimise către memoria internă a calculatorului. Această operație este reprezentată (convențional) prin cuvântul **citește**.

Prin *operația de ieșire* (sau operație de scriere) valorile corespunzătoare datelor de ieșire sunt preluate din memoria internă a calculatorului și sunt transmise către un dispozitiv de ieșire (monitor, imprimantă, dischetă, CD). Această operație este reprezentată (convențional) prin cuvântul **scrie**.

OPERAȚII DE INTRARE – IEȘIRE	
citește id_var ₁ , id_var ₂ ,..., id_var _n	scrie id_var ₁ , id_var ₂ ,..., id_var _n
<i>Exemplu:</i> Se citesc două numere întregi x și y. Să se afișeze suma lor.	
Date de intrare: x, y întreg Date de ieșire: x+y întreg	citește x, y scrie x+y

Tabelul 2. Operații de intrare/ieșire

4.2. Operații de atribuire

Prin operația de atribuire, o variabilă primește o valoare dată, valoarea unei alte variabile sau valoarea unei expresii. Operația de atribuire este reprezentată prin operatorul de atribuire (săgeată) \leftarrow .

variabilă \leftarrow valoare dată			variabilă \leftarrow variabila1	variabilă \leftarrow expresie
Exemple:			Fie $m = 3$ și $n = 5$ două variabile întregi. Să se interschimbe cele două valori ($m=5$ și $n=3$)	Se citesc trei numere reale. Să se calculeze și să se afișeze media lor aritmetică
a real $a \leftarrow 4.67$	c șir de caractere $c \leftarrow \text{'atribuire'}$	x întreg $x \leftarrow 67$	m, n, a întreg $m \leftarrow 3$ $n \leftarrow 5$ $a \leftarrow m$ $m \leftarrow n$ $n \leftarrow a$ scrie m, n	a,b,c, ma real citește a, b, c $ma \leftarrow (a+b+c)/3$ scrie ma

Tabelul 3. Operații de atribuire

4.3. Operații aritmetice, relaționale și logice

Datele de intrare sunt prelucrate cu ajutorul unor operații aritmetice, relaționale sau logice. Aceste operații se efectuează în expresii.

Pentru evaluarea expresiilor, este necesară cunoașterea priorității, pentru fiecare tip de operand care face parte din expresie.

În Tabelul 4 sunt reprezentați operatorii generali utilizați în evaluarea unor expresii, în ordinea descrescătoare a priorităților.

OPERATORI ARITMETICI	OPERATORI RELAȚIONALI	OPERATORI LOGICI																									
<p><i>General:</i> Operanți numerici (întregi, reali). <i>Particular:</i> date calendaristice. <i>Observații:</i> Rezultatul evaluării unei expresii aritmetice este tot numeric. Operatorii aritmetici sunt binari (se aplică pentru doi operanți).</p>	<p><i>General:</i> Operanți de același tip (numerici, logici, caractere, date calendaristice). <i>Observații:</i> Valoarea unei expresii relaționale este de tip logic (adevărat sau fals). Operatorii relaționali sunt binari.</p>	<p><i>General:</i> Operanți logici (expresii relaționale). <i>Observații:</i> Valoarea unei expresii logice este de tip logic. Operatorii logici pot fi binari (conjuncția <i>and</i>, disjuncția <i>or</i>) sau unari (negația: <i>not</i>). Valorile de adevăr sunt: <i>adevărat</i> (A) <i>fals</i> (F)</p>																									
<p><i>Operatori aritmetici multiplicativi:</i> înmulțire * împărțire / Câtul împărțirii întregi div Restul împărțirii întregi mod <i>Operatori aritmetici aditivi:</i> adunare + scădere -</p>	<p>Operatorul de egalitate = Operatorul diferit < > sau ≠ Operatorul mai mic < Operatorul mai mic sau egal < = Operatorul mai mare > Operatorul mai mare sau egal ≥</p>	<p>Operatorul pentru negație not Operatorul conjuncție and Operatorul disjuncție or Regulile de compunere a operatorilor logici:</p> <table border="1"> <thead> <tr> <th>a</th> <th>b</th> <th>not a</th> <th>a and b</th> <th>a or b</th> </tr> </thead> <tbody> <tr> <td>F</td> <td>F</td> <td>A</td> <td>F</td> <td>F</td> </tr> <tr> <td>F</td> <td>A</td> <td>A</td> <td>F</td> <td>A</td> </tr> <tr> <td>A</td> <td>F</td> <td>F</td> <td>F</td> <td>A</td> </tr> <tr> <td>A</td> <td>A</td> <td>F</td> <td>A</td> <td>A</td> </tr> </tbody> </table>	a	b	not a	a and b	a or b	F	F	A	F	F	F	A	A	F	A	A	F	F	F	A	A	A	F	A	A
a	b	not a	a and b	a or b																							
F	F	A	F	F																							
F	A	A	F	A																							
A	F	F	F	A																							
A	A	F	A	A																							

Tabelul 4

Evaluarea expresiilor

Dacă expresia nu conține paranteze rotunde, atunci ea este evaluată de la stânga spre dreapta, în ordinea descrescătoare a priorității operatorilor. Prioritatea operatorilor poate fi modificată prin includerea unor operații între paranteze rotunde.

Prioritatea operatorilor (în ordine descrescătoare) este redată în Tabelul 5.

Prioritate	Operatori	Simbol	Asociativitate
0	Paranteze	()	De la stânga la dreapta
1	Negația logică	not	De la dreapta la stânga
2	Aritmetici multiplicativi	* , / , div , mod	De la stânga la dreapta
3	Aritmetici aditivi	+ , -	De la stânga la dreapta
4	Relaționali	= , < , > , ≠ , <= , < , > , >=	De la stânga la dreapta
5	Conjuncția logică	and	De la stânga la dreapta
6	Disjuncția logică	or	De la stânga la dreapta

Tabelul 5

4.4. Operații de decizie

În foarte multe situații reale, executarea unor operații este condiționată de producerea unor evenimente. Dirijarea prelucrărilor în funcție de evenimente se face prin operații de decizie.

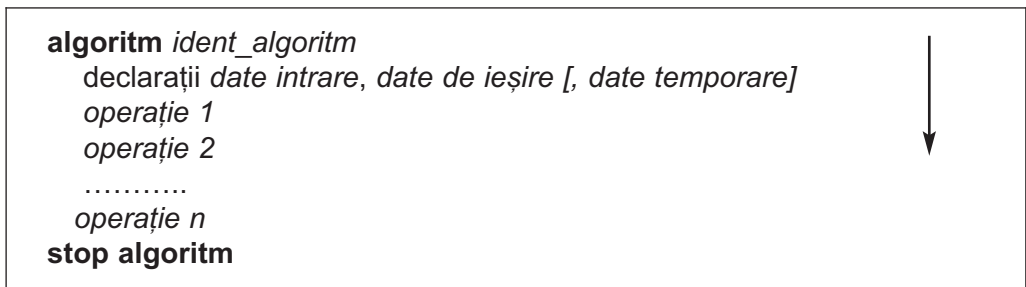
Exemplu:

Situația școlară este influențată de numărul de corigențe:

- Dacă elevul are una sau două corigențe, atunci la sfârșitul anului școlar el este declarat *corigent*.
- Dacă elevul are trei sau mai multe corigențe, atunci el este declarat la sfârșitul anului școlar *repetent*.
- Altfel, elevul este declarat *promovat*.

5. Prelucrări structurate (noțiuni de programare)

În elaborarea algoritmilor se efectuează operații de intrare/ieșire, aritmetice, relaționale, logice, de decizie. Executarea acestor operații se efectuează într-o ordine logică, de sus în jos (top-down), ca în figura de mai jos:



În cadrul unui algoritm, organizarea prelucrărilor se face astfel încât să fie respectate principiile programării structurate:

1. Principiul modularizării

Pentru reprezentarea algoritmică a unei probleme complexe, aceasta poate fi descompusă în subprobleme relativ independente, pentru fiecare subproblemă construindu-se subalgoritmi mai simpli. Fiecare subalgoritm cuprinde un set de prelucrări (operații) specifice și este relativ independent de ceilalți subalgoritmi. Subalgoritmii comunică între ei prin intermediul unor parametri. Ordinea de parcurgere a subalgoritmilor este stabilită prin apeluri din subalgoritmul principal (de bază).

Avantajele oferite de acest principiu:

- fiecare modul (subalgoritm) poate fi elaborat, testat, modificat, depanat independent de celelalte module (subalgoritmi);
- modificarea unui modul (subalgoritm) nu afectează celelalte module (subalgoritmi).

2. Principiul structurării datelor și a prelucrărilor

Pentru prelucrarea datelor este necesară uneori gruparea acestora după anumite criterii. Clasele de date astfel obținute se numesc *structuri de date*.

Operațiile utilizate într-un algoritm pot fi grupate sau prelucrate în diferite forme numite *structuri de control*.

Orice prelucrare poate fi descrisă prin combinarea a trei tipuri de structuri de control fundamentale:

- *structuri liniare*,
- *structuri alternative*,
- *structuri repetitive*.

Pentru reprezentarea structurilor de control fundamentale și a operațiilor elementare într-un algoritm, se folosește un limbaj convențional numit *pseudocod*. Acest limbaj codifică operațiile și structurile fundamentale, permițând transpunerea algoritmului în orice limbaj de programare.

5.1. Structuri liniare (secvențiale)

Structura liniară (secvențială) cuprinde operații de intrare (citire), operații de atribuire, operații aritmetice, operații de ieșire (scriere) executate în ordinea „de sus în jos”. Disponerea operațiilor respectă logica problemei. Operațiile dintr-o structură liniară se execută necondiționat, o singură dată.

```
algoritm modularizat
  subalgoritm sa1
    operație 1.1
    operație 1.2
    .....
  stop subalgoritm sa1

  subalgoritm sa2
    operație 2.1
    operație 2.2
    .....
  stop subalgoritm sa2
.....
subalgoritm principal
  operație 1
  operație 2
  .....
  apel sa 1
  apel sa 2
  .....
stop subalgoritm principal

stop algoritm modularizat
```

APLICAȚII REZOLVATE:

1. Fie **a** și **b** două numere reale strict pozitive reprezentând laturile unui dreptunghi. Să se scrie un algoritm, în pseudocod, pentru calcul și afișarea perimetrului și ariei dreptunghiului.

Date de intrare : a, b real Date de ieșire : p real // perimetrul dreptunghiului S real // aria dreptunghiului	algoritm ex1 citește a, b P ← 2*(a+b) S ← a*b scrie p, S stop algoritm ex1
--	---

2. Andrei observă că rezerva sa de CD-uri s-a epuizat. El își propune ca, din economiile sale, să folosească S lei pentru CD-uri noi. Un CD costă x lei. Să se scrie un algoritm, în pseudocod, pentru calcul și afișarea numărului maxim de CD-uri pe care le poate cumpăra Andrei și afișarea sumei rămase după cumpărarea CD-urilor.

Date de intrare : S, x întreg Date de ieșire : n întreg // numărul maxim de CD-uri rest întreg // suma de bani rămasă	algoritm ex2 citește S, x n ← S div x rest ← S - n*x scrie n, rest stop algoritm ex2
---	---

TEME:

1. a) Ce va afișa algoritmul următor, dacă se citesc valorile 12 și 29 ?

b) Propuneți o pereche de valori pentru x și y, astfel încât algoritmul să afișeze valorile 56 și 34.

Date de intrare : x, y întreg Date de ieșire : x, y întreg	algoritm tema1 citește x citește y x ← x + y y ← x - y x ← x - y scrie x scrie y stop algoritm tema1
---	--

2. Se consideră următorul algoritm reprezentat în pseudocod. Determinați ce valoare va afișa algoritmul, dacă se citesc valorile 24, 123 și 67. Care dintre variantele de răspuns propuse este corectă ?

Date de intrare : x, y, z întreg Date de ieșire : s întreg	algoritm tema2 citește x citește y citește z $s \leftarrow 0$ $s \leftarrow s + x \text{ mod } 10$ $s \leftarrow s + y \text{ mod } 10$ $s \leftarrow s + z \text{ mod } 10$ scrie s stop algoritm tema2
Variante de răspuns: a) 9; b) 14; c) 84; d) 20.	

3. Se citesc trei numere reale **a**, **b** și **c** care reprezintă laturile unui triunghi. Scrieți un algoritm în pseudocod, care să calculeze și să afișeze perimetrul și aria triunghiului.

4. Alexandra lucrează la o firmă de publicitate și a primit în anul 2005 un salariu lunar de 1.200 RON. În lunile aprilie, mai și iunie a avut o mărire salarială de 15%, în lunile septembrie și octombrie a avut o mărire de 20%, iar în luna decembrie a mai primit o primă de 1.000 RON. Ajutați-o să-și cunoască câștigul realizat în anul trecut și venitul mediu lunar, scriind un algoritm în limbaj pseudocod.

5. Se consideră următorul algoritm reprezentat în pseudocod. Care variantă de răspuns reprezintă valorile afișate de algoritm ?

Date de intrare: u, v real Date de ieșire : u, v real	algoritm tema5 $u \leftarrow 25$ $t \leftarrow 4$ $u \leftarrow u / 2 * t$ $t \leftarrow t + u / (2 * t)$ $u \leftarrow u + t$ $t \leftarrow t + u$ scrie u scrie t stop algoritm tema5
Variante de răspuns: a) 60.25 60.25; b) 8.78125 13.31350. c) 60.25 70.50; d) 8.78125 8.78125.	

5.2. Structuri alternative

Structura alternativă cuprinde o operație de decizie și două secvențe de operații, dintre care se execută doar una, în funcție de valoarea de adevăr a condiției logice.

Acestă structură se reprezintă și se definește în pseudocod ca în figura 1.

Figura 1: Structura alternativă

	<p>dacă (expresie logică) atunci secvența1 de operații altfel secvența2 de operații</p> <p>sfârșit dacă</p>
<p>Reprezentarea structurii alternative</p>	<p>Sintaxa structurii alternative în pseudocod</p>

Mecanismul de execuție a structurii alternative:

Pasul 1: Se evaluează expresia logică.

Pasul 2: Dacă valoarea expresiei este *adevărat* atunci se execută secvența1; dacă valoarea expresiei este *fals* atunci se execută secvența 2.

Există situații în care structura alternativă solicită executarea unei singure secvențe de operații pentru cazul în care valoarea de adevăr a expresiei logice este adevărat. În acest caz, din structură lipsește secvența de operații de pe ramura **altfel**; structura se numește *pseudoalternativă* și se reprezintă astfel:

Figura 2: Structura pseudoalternativă

	<p>dacă (expresie logică) atunci secvența de operații</p> <p>sfârșit dacă</p>
<p>Reprezentarea structurii pseudoalternative</p>	<p>Sintaxa structurii alternative în pseudocod</p>

Secvențele de operații de pe oricare ramură pot avea subordonate alte structuri alternative. Se obțin structuri alternative incluse, denumite și *structuri alternative imbricate*.

```

dacă (expresie logică1) atunci
    dacă (expresie logică2) atunci
        secvența1 de operații
    altfel
        secvența2 de operații
    sfârșit dacă
altfel
    secvența3 de operații
sfârșit dacă

```

APLICAȚII REZOLVATE:

1. Se citesc două numere n , k întregi nenule. Să se scrie un algoritm prin care să se verifice dacă n este divizibil cu k și să se afișeze un mesaj corespunzător.

<p>Date de intrare: n, k întreg Date de ieșire: un mesaj</p>	<pre> algoritm ex1 citește n citește k dacă ($n \bmod k = 0$) atunci scrie 'numerele sunt divizibile' altfel scrie 'numerele nu sunt divizibile' sfârșit dacă stop algoritm ex1 </pre>
--	---

2. Patru prieteni organizează un miniconcurs de șah. După ce s-au jucat toate partidele planificate, cei patru au acumulat următoarele punctaje: Andrei p_1 puncte, Cristian p_2 puncte, Cătălina p_3 puncte, iar Cosmin p_4 puncte (punctajele sunt numere naturale). Să se scrie un algoritm prin care să se determine și să se afișeze punctajul maxim.

<p>Date de intrare: p_1, p_2, p_3, p_4 întreg Date de ieșire: punctajul maxim</p>	<pre> algoritm ex2 citește p_1, p_2, p_3, p_4 maxim $\leftarrow p_1$ dacă ($\text{max} > p_2$) atunci max $\leftarrow p_2$ altfel dacă ($\text{max} > p_3$) atunci maxim $\leftarrow p_3$ altfel maxim $\leftarrow p_4$ sfârșit dacă sfârșit dacă scrie 'Punctajul obținut de câștigător este ', maxim stop algoritm ex2 </pre>
--	---

TEMĂ

Un angajat are un salariu de bază; în funcție de vechimea în câmpul muncii, salariul său crește cu un anumit procent (în tabelul alăturat sunt precizate procentele acordate în funcție de vechime).

a) Se cunosc vechimea și salariul de bază. Să se scrie un algoritm prin care să se afișeze salariul angajatului.

b) Completați coloana Salariu din tabelul alăturat.

Vechime ani	Procent	Salariu bază RON	Salariu RON
1-5	10%		
6-8	12%	1000	1120
9-12	15%		
13-15	18%		
16-20	20%		
>20	25%		

5.3. Structuri repetitive

În rezolvarea anumitor probleme, anumite operații sau prelucrări se repetă de un număr cunoscut de ori sau până se îndeplinește o condiție de oprire.

Exemple: (1) calculul numărului de absențe motivate și nemotivate ale tuturor elevilor dintr-o clasă (suma se calculează de un număr cunoscut de ori = numărul de elevi din clasă); (2) citirea, de la tastatură, a notei unui elev; operația se repetă până când valoarea introdusă corespunde unei note (aparține intervalului $[1, 10]$).

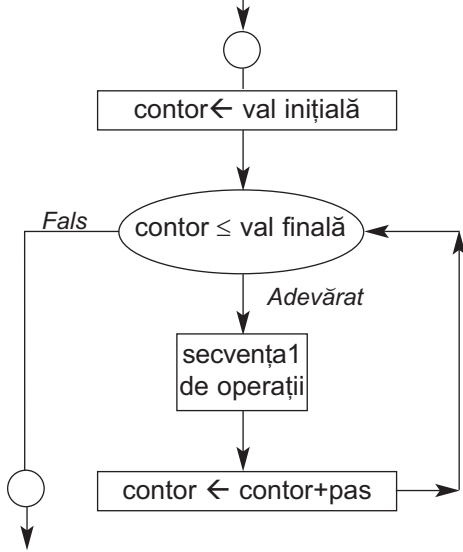
5.3.1. Structuri repetitive cu număr cunoscut de pași (cu contor)

Pentru reprezentarea operațiilor al căror număr de repetiții este cunoscut, se utilizează structuri repetitive cu contor. Contorul este o variabilă în care se numără operațiile executate. Contorul pornește de la o valoare inițială și se modifică cu un pas până la o valoare finală.

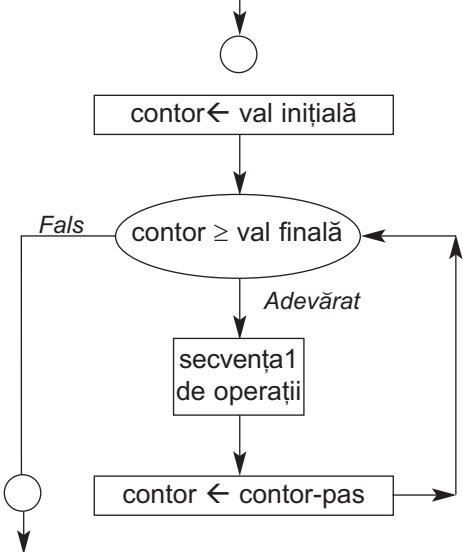
Dacă valoarea inițială este mai mică decât valoarea finală, structura repetitivă are contor crescător.

Dacă valoarea inițială este mai mare decât valoarea finală, structura repetitivă are contor descrescător.

Reprezentarea structurii repetitive cu număr cunoscut de pași (cu contor crescător) – varianta 1



Reprezentarea structurii repetitive cu număr cunoscut de pași (contor descrescător) – varianta 2



Sintaxa structurii repetitive cu număr cunoscut de pași în pseudocod

pentru contor ← val_in, val_fin, pas
execută
 secvența de operații
sfârșit pentru

Mecanismul de execuție a structurii repetitive cu contor crescător:

Pasul 1: Se inițializează variabila contor cu o valoare (expresie) inițială.

Pasul 2: Se compară variabila contor cu valoarea finală (expresie). Dacă contorul este mai mic sau egal cu valoarea finală, atunci se trece la Pasul 3, altfel se trece la operația din secvența care urmează după structura repetitivă.

Pasul 3: Se execută secvența de operații și se trece la Pasul 4.

Pasul 4: Variabila contor este incrementată cu o valoare constantă **pas** (contor ← contor + pas), care în general are valoarea 1 (contor ← contor + 1) și se reia Pasul 2.

APLICAȚII REZOLVATE:

1. Se citește un număr **n** natural nenul. Să se scrie un algoritm pentru:

- a) afișarea primelor **n** numere naturale în ordine descrescătoare;
- b) afișarea primelor **n** numere naturale impare în ordine crescătoare.

Pentru n = 10	
a) Se va afișa 10 9 8 7 6 5 4 3 2 1	b) Se va afișa 1 3 5 7 9

Rezolvare:

<p>Date de intrare: n natural Date de ieșire: mesaje Date intermediare: i (variabila contor)</p>	<pre> algoritm ex1 citește n pentru i ← n, 1, -1 execută // pas decrementare = -1 scrie i , ‘ ‘ sfârșit pentru pentru i ← 1, n, 2 execută // pas incrementare = 2 scrie i , ‘ ‘ sfârșit pentru stop algoritm ex1 </pre>
---	---

2. Se citesc trei numere naturale **a**, **b** ($a < b$) și **d** ($d > 0$). Să se afișeze toate numerele din intervalul **[a, b]** divizibile cu valoarea **d** și câte numere cu această proprietate s-au determinat.

Exemplu:

Pentru $a = 4$, $b = 29$ și $d = 7$

Se va afișa șirul de valori : 7, 14, 21

Numărul de elemente = 3

Rezolvare:

<p>Date de intrare: a, b, d întreg Date de ieșire: numerele divizibile cu d (dacă există) c numărul de elemente divizibile cu d Date intermediare: i (variabila contor)</p>	<pre> algoritm ex2 citește a, b, d c ← 0 pentru i ← a , b, 1 execută // pas incrementare = 1 dacă (i mod d == 0) atunci scrie d c ← c+1 sfârșit dacă sfârșit pentru scrie c sfârșit pentru stop algoritm ex2 </pre>
---	---

TEME

1. Explicați mecanismul structurii repetitive cu contor descrescător.
2. Să se scrie un algoritm în pseudocod pentru afișarea triunghiului de numere din caseta alăturată, unde **n** este un număr natural nenul citit.
3. Se citesc **n** numere întregi. Să se determine și să se afișeze:
 - a) câte dintre ele sunt pare;
 - b) suma elementelor pozitive;
 - c) valoarea maximă.
4. Într-o clasă sunt 28 de elevi. Să se scrie un algoritm în pseudocod pentru calculul și afișarea numărului de absențe motivate, numărul de absențe nemotivate ale fiecărui elev, numărul total de absențe motivate, valoarea medie a absențelor nemotivate.

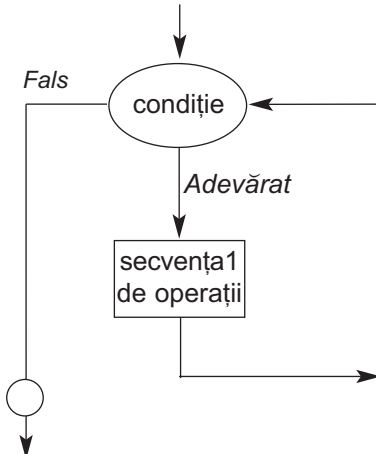
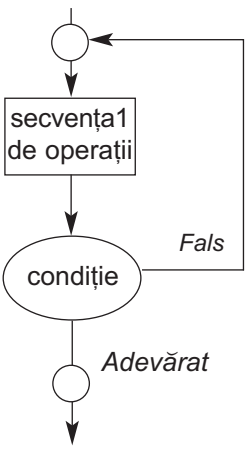
1
1 2
1 2 3
.....
1 2 3 ... n

5. Se citește un număr n natural nenul. Să se calculeze și să se afișeze valoarea următoarei expresii:

$$E = \left(1 - \frac{1}{2^2}\right) \cdot \left(1 - \frac{1}{3^2}\right) \cdots \left(1 - \frac{1}{n^2}\right)$$

5.3.2. Structuri repetitive condiționate

Pentru situațiile în care repetarea unei secvențe de operații este controlată de respectarea unei condiții, se pot folosi structurile **cât timp–repetă** sau **repetă–până când**.

<p>Reprezentarea structurii repetitive cu testare inițială</p> 	<p>Reprezentarea structurii repetitive cu testare finală</p> 	
<p>Sintaxa structurii repetitive cu testare inițială în pseudocod</p>	<p>Sintaxa structurii repetitive cu testare finală în pseudocod</p>	
<p>cât timp condiție execută secvența de operație sfârșit cât timp</p>	<p>repetă secvența de operații până când condiție</p>	<p>execută secvența de operații cât timp condiție</p>

Structurile repetitive condiționate sunt complementare: trecerea de la un tip de structură la altul se face prin negarea condiției.

Mecanismul de execuție a structurii repetitive cu testare inițială (**cât timp** execută):

Pasul 1: Se testează condiția. Dacă este îndeplinită condiția (valoarea expresiei condiționale este *adevărat*), atunci se trece la Pasul 2, altfel se iese din structura repetitivă **cât timp**.

Pasul 2: Se execută secvența de operații.

Mecanismul de execuție a structurii repetitive cu testare finală (**repetă până când**):

Pasul 1: Se execută secvența de operații.

Pasul 2: Se testează condiția. Dacă nu este îndeplinită condiția (valoarea expresiei condiționale este *fals*) atunci se reia Pasul 2, altfel se iese din structura repetitivă.

Exemple:

Exemplul 1. Să se determine suma cifrelor unui număr natural nenul n .

Suma cifrelor unui număr natural nenul n			
Date de intrare: n număr natural nenul	algoritm suma_1 citește n $s \leftarrow 0$ cât timp ($n > 0$) execută	algoritm suma_2 citește n $s \leftarrow 0$ execută	algoritm suma_3 citește n $s \leftarrow 0$ repetă
Date de ieșire: s număr natural nenul suma cifrelor	$s \leftarrow s + n \bmod 10$ $n \leftarrow n \div 10$ sfârșit cât timp scrie s sfârșit algoritm suma_1	$s \leftarrow s + n \bmod 10$ $n \leftarrow n \div 10$ cât timp ($n > 0$) scrie s sfârșit algoritm suma_2	$s \leftarrow s + n \bmod 10$ $n \leftarrow n \div 10$ până când ($n = 0$) scrie s sfârșit algoritm suma_3

Exemplul 2. Să se calculeze și să se afișeze suma primelor n numere naturale.

Deși se cunoaște numărul de elemente care trebuie prelucrate, în rezolvarea care urmează se va trata echivalența dintre structura repetitivă cu contor și structurile repetitive condiționale.

Suma primelor n numere naturale			
Date de intrare: n număr natural nenul	algoritm suma_1 citește n $s \leftarrow 0$ $i \leftarrow 1$ cât timp ($i \leq n$) execută	algoritm suma_2 citește n $s \leftarrow 0$ $i \leftarrow 1$ execută	algoritm suma_3 citește n $s \leftarrow 0$ $i \leftarrow 1$ repetă
Date de ieșire: s număr natural nenul suma	$s \leftarrow s + i$ $i \leftarrow i + 1$ sfârșit cât timp scrie s sfârșit algoritm suma_1	$s \leftarrow s + i$ $i \leftarrow i + 1$ cât timp ($i \leq n$) scrie s sfârșit algoritm suma_2	$s \leftarrow s + i$ $i \leftarrow i + 1$ până când ($i > n$) scrie s sfârșit algoritm suma_3
Date intermediare: i număr natural, variabilă contor			

TEME

1. Se citește un număr natural nenul. Să se determine și să se afișeze cifrele pare ale acestui număr.
2. Se citesc două numere întregi a și b . Să se afișeze numerele din intervalul format de valorile a și b .
3. Se citesc numere întregi până când se întâlnește 0. Să se calculeze și să se afișeze media aritmetică a numerelor strict negative.

6. Prelucrarea grupurilor de date

6.1. Organizarea grupurilor de date

- La un centru de meteorologie se fac măsurători zilnice ale temperaturii și vitezei vântului. La sfârșitul fiecărei luni, se cere ziua cea mai călduroasă, ziua cu temperatura cea mai scăzută, câte zile au avut temperaturi sub temperatura medie, viteza medie a vântului.

Pentru determinarea acestor valori, sunt necesare prelucrări care se pot aplica grupului de date *temperaturi*. Grupurile de date pot fi păstrate în memoria internă sub formă de *tablouri unidimensionale* de date sau în memoria externă sub formă de fișiere de date.

Un *tablou unidimensional* (liniar) este caracterizat printr-un identificator (nume) și capacitatea tabloului (numărul maxim de elemente); fiecare element din tablou se identifică prin numele tabloului și adresa elementului în tablou numită poziție sau indice. Elementele unui tablou au același tip de dată.

Exemple:

- tabloul t cu 31 de elemente de tip întreg care reprezintă temperaturile zilnice; $t[10]$ – temperatura din ziua a 10-a;
- tabloul v tot cu 31 de elemente de tip real care reprezintă viteza vântului; $v[5]$ – viteza vântului în ziua a 5-a.

- *Datele dintr-o agendă personală reprezintă un grup de valori despre persoane; fiecare persoană este descrisă prin aceleași caracteristici: nume, adresă, data nașterii, număr de telefon, adresă e-mail. Informațiile despre o persoană formează un grup neomegen de date numit articol sau înregistrare. Agenda poate fi reprezentată printr-un tablou unidimensional cu mai multe înregistrări persoană.*

O înregistrare se definește printr-un identificator (în exemplul nostru, *persoana*) și două sau mai multe câmpuri (în exemplul dat, caracteristicile unei persoane).

Din exemplele prezentate, distingem două categorii de grupuri de date: grupuri de *date omogene* (tablouri) și grupuri de date neomogene (articol); mai multe articole pot fi organizate într-un tablou (fig. 3).

TIPURI DE DATE STRUCTURATE							
OMOGENE	NEOMOGENE						
Tabloul liniar cu n elemente	Înregistrarea persoană						
<p style="text-align: center;">T</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">2</td> <td style="text-align: center;">3</td> <td style="text-align: center;">.....</td> <td style="text-align: center;">n-1</td> <td style="text-align: center;">n</td> </tr> </table>	1	2	3	n-1	n	<p>Persoană</p> <p>Cod persoană</p> <p>Nume</p> <p>Prenume</p> <p>.....</p> <p>Data nașterii</p> <p>.....</p>
1	2	3	n-1	n		

Figura 3: Structuri de date

În continuare, prin grup de date vom face referire la tablouri de date.

6.2. Citirea și afișarea grupurilor de date

Grupurile de date pot fi prelucrate astfel încât rezultatele să caracterizeze întregul grup: valoarea medie a grupului (media generală a clasei), așezarea elementelor grupului după o anumită ordine (catalogul clasei – elevii sunt scriși în ordine alfabetică).

Cele mai simple prelucrări asigură memorarea și afișarea datelor grupate.

Exemplu:

Se citesc cele 31 de temperaturi înregistrate în registrul centrului meteorologic și apoi se afișează pe ecran.

Rezolvare:

Date de intrare: n număr natural $(n=31) (t_i \in \mathbb{Z})_{i=1,n}$ Date de ieșire: $(t_i \in \mathbb{Z})_{i=1,n}$ Date intermediare: i (variabila contor)	<pre> algoritm citire_afișare citește n pentru $i \leftarrow 1, n$ execută citește t_i sfârșit pentru pentru $i \leftarrow 1, n$ execută scrie t_i sfârșit pentru stop algoritm citire_afișare </pre>
--	---

TEME

1. Scrieți secvența pentru citirea și afișarea unui grup de date – mediile la Informatică ale tuturor elevilor din clasa voastră. Folosiți o structură repetitivă condiționată.

2. Scrieți secvența pentru citirea și afișarea unui grup de date: înălțimile tuturor băieților din școala voastră; determinați înălțimea medie a grupului de băieți.

6.3. Ordonarea grupurilor de date

Ordonarea grupurilor de date (**sortarea**) se face prin schimbarea poziției elementelor din grup (interschimb), astfel încât toate elementele să respecte un criteriu specificat numit **criteriu de sortare**. Dacă toate elementele grupului sunt așezate astfel încât valoarea oricărui element i să fie mai mică decât valoarea elementului $i+1$, grupul este sortat crescător.

Ordonarea alfabetică este crescătoare.

Dacă toate elementele grupului sunt așezate astfel încât valoarea oricărui element i să fie mai mare decât valoarea elementului $i+1$, grupul este sortat descrescător. Ordonarea candidaților după media la examen este descrescătoare.

Exemplu de problemă care necesită ordonarea unui grup de date

La fiecare început de sezon, clubul de baschet 'Astra' face noi selecții; cei n candidați sunt trimiși la cabinetul medical pentru a li se măsura înălțimea.

Să se afișeze lista candidaților după înălțime.

Pentru rezolvarea problemei, înălțimile vor fi memorate într-un tablou h .

Pentru a ordona elementele unui tablou, se cunosc mai mulți algoritmi de sortare; dintre aceștia vom prezenta: *sortarea prin interschimbări directe*, *bubble-sort*, *sortarea prin selecție*.

6.3.1. Algoritmul de sortare prin interschimbări directe

Înainte de ordonare, elementele tabloului h sunt așezate în ordinea din figura 4.

Ordonarea crescătoare a candidaților după înălțime necesită următoarele prelucrări:

Pasul 1: Se compară primul element (pivot) h_1 cu elementele situate la dreapta sa (h_2, h_3, \dots, h_n). Dacă h_1 este mai mare decât un element h_j , $j=2, n$ atunci cele două elemente se interschimbă ($h_1 \leftrightarrow h_j$). După acest șir de $n-1$ comparații pe prima poziție a tabloului se va afla valoarea minimă din șir.

Pasul 2: Se compară al doilea element (pivot) h_2 cu elementele situate la dreapta sa (h_3, h_4, \dots, h_n). Dacă h_2 este mai mare decât un element h_j , $j=3, n$ atunci cele două elemente se interschimbă ($h_2 \leftrightarrow h_j$). După acest șir de $n-2$ comparații pe prima poziție a tabloului se va afla valoarea minimă a elementelor situate din tablou pe pozițiile 2,3,..., n .

Pasul $n-1$: Se compară penultimul element (pivot) h_{n-1} cu ultimul element h_n . Dacă h_{n-1} este mai mare decât h_n , atunci cele două elemente se interschimbă ($h_{n-1} \leftrightarrow h_n$).

După acest pas șirul este ordonat crescător.

Indice Pas	Tabloul h								
		2	3	4	5	6	7	8	9
Inițial	1.70	1.85	1.75	1.65	1.85	1.90	1.80	1.73	1.95
Pas 1	↑	↑	↑						
	1.65	1.85	1.75	1.70	1.85	1.90	1.80	1.73	1.95
				↑	↑	↑	↑	↑	
Pas 2	1.65	1.85	1.75	1.70	1.85	1.90	1.80	1.73	1.95
	↑								
	1.65	1.75	1.85	1.70	1.85	1.90	1.80	1.73	1.95
		↑							
Pas n-1	1.65	1.70	1.73	1.75	1.80	1.85	1.85	1.90	1.95
	↑								
	1.65	1.70	1.73	1.75	1.80	1.85	1.85	1.90	1.95

Figura 4: Sortarea datelor

Reprezentarea algoritmului

<p>Date de intrare : n număr natural $(h_i \in \mathbb{R})_{i=1,n}$ valorile sunt așezate într-o ordine oarecare Date de ieșire : $(h_i \in \mathbb{R})_{i=1,n}$ Valorile vor fi așezate crescător Date intermediare: i, j variabile contor aux număr real</p>	<p>Algoritm ordonare1 pentru $i \leftarrow 1, n-1$ execută pentru $j \leftarrow i+1, n$ execută dacă $h_i > h_j$ atunci</p> <p style="text-align: right;">$aux \leftarrow h_i$ $h_i \leftarrow h_j$ $h_j \leftarrow aux$</p> <p>sfârșit dacă sfârșit pentru sfârșit pentru sfârșit algoritm ordonare1</p>
---	---

6.3.2. Algoritm de sortare prin metoda bulelor (bubble-sort)

Pentru descrierea algoritmului vom folosi același tablou h cu n elemente și o variabilă cu rol de semafor (variabila are doar două valori, cu semnificația: 0 - tabloul nu este ordonat și 1 - tabloul este ordonat); vezi figura 5.

Pasul 1: Variabila semafor se inițializează cu valoarea 0.

Pasul 2: Se compară două elemente consecutive în șir (se compară primul element h_1 cu al doilea h_2 , h_2 cu h_3 , ..., h_{n-1} cu h_n) până când se parcurge tot

tabloul. Dacă h_i este mai mare decât h_{i+1} , $i=1, n-1$ atunci cele două elemente se interschimbă ($h_i \leftrightarrow h_{i+1}$) și semaforul devine 1.

Pasul 3: Dacă la Pasul 2, după o parcurgere a tabloului, se face cel puțin o interschimbare (semaforul = 1) atunci se reiau Pașii 1 și 2.

Dacă la Pasul 2 nu se face nicio interschimbare (semaforul = 0) atunci tabloul are toate elementele ordonate crescător.

Numele metodei este foarte sugestiv față de modul în care se deplasează elementele grupului, în timpul sortării: valorile mici (bulele) sunt împinse în față.

Figura 5: Bubble-Sort

Indice Pas	Tabloul h								
	1	2	3	4	5	6	7	8	9
Inițial	1.70	1.85	1.65	1.75	1.85	1.90	1.80	1.73	1.95
Parcurerea 1	↑ ↑								
	sem=0								
	1.70	1.75	1.85	1.65	1.85	1.90	1.80	1.73	1.95
	sem=1								
	1.70	1.75	1.65	1.85	1.85	1.90	1.80	1.73	1.95
	sem=1								
	1.70	1.75	1.65	1.85	1.85	1.80	1.90	1.73	1.95
sem=1									
sem=1									
sem=1									
sem=1									
sem=1									
Parcurerea 2	1.70	1.75	1.65	1.85	1.85	1.80	1.73	1.90	1.95
	sem=0								
	1.70	1.65	1.75	1.85	1.85	1.80	1.73	1.90	1.95
	sem=1 In								
	1.70	1.65	1.75	1.85	1.80	1.85	1.73	1.90	1.95
	Sem=1								
Sem=1									
Sem=1									
...								
Ultima parcurgere	1.65	1.70	1.73	1.75	1.80	1.85	1.85	1.90	1.95
	Sem=0								
	1.65	1.70	1.73	1.75	1.80	1.85	1.85	1.90	1.95

Reprezentarea algoritmului

<p>Date de intrare : n număr natural $(h_i \in \mathbb{R})_{i=1,n}$</p> <p>Date de ieșire : $(h_i \in \mathbb{R})_{i=1,n}$</p> <p>Date intermediare: i (variabila contor) aux număr real sem variabilă semafor</p>	<p>Algoritm <i>ordonare2_2</i> repetă semafor $\leftarrow 0$ pentru $i \leftarrow 1, n-1$ execută dacă $h_i > h_{i+1}$ atunci aux $\leftarrow h_i$ $h_i \leftarrow h_j$ $h_j \leftarrow aux$ semafor $\leftarrow 1$</p> <p> sfârșit dacă sfârșit pentru până când (semafor=0) sfârșit algoritm <i>ordonare2_2</i></p>
--	---

6.3.3. Algoritm de sortare prin selecție

La baza acestui algoritm stă determinarea elementului minim (*min*) din subșiruri de lungimi descrescătoare ($n, n-1, \dots, 1$) și interschimbarea acestuia cu elementul aflat pe prima poziție p a șirului prelucrat. Pentru realizarea corectă a interschimbării, se reține și poziția minimului găsit în șirul prelucrat.

Pasul 1: Se inițializează **min** cu h_1 , se reține în variabila m poziția minimului, respectiv $m=1$; se compară elementele ($h_i, i=2,n$) aflate la dreapta elementului h_1 cu valoarea **min**. Dacă se găsește o valoare mai mică, atunci **min** va păstra valoarea acestui element și **poz_min** va prelua indicele acestuia. După parcurgerea completă, $h_{\text{poz}_m} \leftarrow h_1$ și $h_1 \leftarrow \text{min}$.

Pasul k: Se inițializează **min** cu h_k , se reține în variabila m poziția minimului, respectiv $m=k$; se compară elementele ($h_i, i=k+1,n$) aflate la dreapta elementului h_k cu valoarea **min**. Dacă se găsește o valoare mai mică, atunci **min** va păstra valoarea acestui element și **poz_min** va prelua indicele acestuia. După parcurgerea completă, $h_{\text{poz}_m} \leftarrow h_k$ și $h_k \leftarrow \text{min}$.

Reprezentarea algoritmului:

<p>Date de intrare : n număr natural $(h_i \in \mathbb{R})_{i=1,n}$</p> <p>Date de ieșire : $(h_i \in \mathbb{R})_{i=1,n}$</p> <p>Date intermediare: i, j - variabile contor poz_min: număr natural, poziția minimului min variabilă reală</p>	<p>Algoritm <i>ordonare3</i> pentru $i \leftarrow 1, n-1$ execută min $\leftarrow h_i$ poz_m $\leftarrow i$ pentru $j \leftarrow i+1, n$ execută dacă $h_j < \text{min}$ atunci min $\leftarrow h_j$ poz_m $\leftarrow j$</p> <p> sfârșit dacă sfârșit pentru $h_{\text{poz}_m} \leftarrow h_i$ $h_i \leftarrow \text{min}$ sfârșit pentru sfârșit algoritm <i>ordonare3</i></p>
---	--

6.4. Căutarea valorilor într-un grup de date


Căutarea valorilor în grupurile de date este o prelucrare foarte frecventă: se caută un număr de telefon în agendă, se caută ziua în care a fost înregistrată cea mai mare temperatură.

Metoda de căutare se aplică în funcție de relația de ordine dintre elementele grupului (grup ordonat sau neordonat).

6.4.1. Căutarea secvențială

Operația de căutare a primei apariții a unei valori într-un tablou necesită o parcurgere a tabloului, element cu element, începând de la primul către ultimul element. Dacă elementul a fost găsit în tablou, se afișează un mesaj corespunzător: *valoare găsită*; se poate afișa și poziția elementului din tablou. Dacă valoarea nu a fost găsită, se afișează mesajul: *valoare inexistentă*.

Viteza vântului [km/oră] în prima săptămână a lunii mai 2006						
v				n=7 (o săptămână)		
12.34	24.45	34.58	15.90	38.95	18.34	45.25

Dacă
v_m = 38.95 atunci **valoare găsită** pe poziția **i=5** 

Dacă
V_m = 56.80 atunci **valoarea nu a fost găsită** (**i = 8 > n**)

<p>Date de intrare :</p> <p>n număr natural (n=31) $(v_i \in \mathbb{R})_{i=1,n}$ $v_m \in \mathbb{R}$</p> <p>Date de ieșire :</p> <p>un mesaj corespunzător și, după caz, indicele i al elementului din tablou a cărui valoare este egală cu v_m</p> <p>Date intermediare:</p> <p>i (variabila contor)</p>	<p>Algoritm <i>căutare_secv</i> <i>//se consideră citite elementele tabloului v</i> citește v_m i ← 1 cât timp (i ≤ n) și (v_i < v_m) execută i ← i+1 sfârșit cât timp dacă i ≤ n atunci scrie 'valoare găsită pe poziția ', i altfel scrie 'valoarea nu există' sfârșit dacă sfârșit algoritm <i>căutare_secv</i></p>
---	---

TEME:

1. Rescrieți algoritmul de căutare secvențială, astfel încât parcurgerea tabloului să înceapă de la ultimul element către primul.


2. Dacă, după parcurgerea tabloului de la ultimul element către primul, valoarea căutată nu a fost găsită, ce valoare are indicele i:

- a) 1; b) 2; c) n; d) 0.

6.4.2. Căutarea într-un tablou ordonat

Dacă tabloul este ordonat, atunci căutarea unei valori se face mai simplu.

Viteza vântului [km/oră] în prima săptămână a lunii mai 2006						
v				n=7 (o săptămână)		
12.34	15.90	18.34	24.45	34.58	38.95	45.25

Dacă
v_m = 38.95 atunci **valoare găsită** pe poziția **i = 6** 

Dacă
v_m = 56.80 atunci **valoarea nu a fost găsită** (**i = 8 > n**)

Descrierea algoritmului

Presupunem că elementele tabloului sunt ordonate crescător. Se poate executa o parcurgere a tabloului, element cu element, începând cu primul. Valoarea **v_m** căutată se compară doar cu elementele din șir care sunt mai mici sau cel mult egale cu **v_m**. Dacă elementul a fost găsit în tablou, căutarea se termină cu succes (se transmite un mesaj corespunzător *valoare găsită* și indicele elementului din tablou a cărui valoare este egală cu valoarea căutată, **v_m**). Dacă s-a parcurs tabloul și toate elementele sale au fost mai mici strict decât valoarea căutată sau dacă nu au fost parcurse toate elementele tabloului și valoarea elementului curent este mai mare decât valoarea căutată **v_m**, atunci căutarea se încheie fără succes (se transmite un mesaj corespunzător: *valoare inexistentă*).

<p>Date de intrare :</p> <p>n număr natural (n=31) ($v_i \in \mathbb{R}$) $i=1, n$ $v_m \in \mathbb{R}$</p> <p>Date de ieșire:</p> <p>Un mesaj corespunzător</p> <p>După caz indicele i al elementului din tablou a cărui valoare este egală cu v_m</p> <p>Date intermediare:</p> <p>i (variabila contor)</p>	<p>Algoritm <i>căutare_ordonat</i> <i>//se consideră citite elementele tabloului v</i> <i>// se ordonează crescător elementele tabloului</i></p> <p>citește v_m i ← 1 cât timp (i <=n) și (v_i <v_m) execută i ← i+1 sfârșit cât timp dacă (i <=n) atunci dacă (v_m=v_i) atunci scrie 'valoare găsită pe poziția ', i altfel scrie 'valoarea nu există' sfârșit dacă altfel scrie 'valoarea nu există'</p> <p>sfârșit dacă sfârșit algoritm <i>căutare_ordonat</i></p>
---	---

TEME

1. Modificați algoritmul pentru căutarea unei valori într-un grup de date ordonat descrescător.

2. Modificați algoritmul de căutare într-un grup ordonat astfel încât parcurgerea tabloului să înceapă de la ultimul element. Se consideră tabloul ordonat crescător.

6.4.3. Căutarea binară

Pentru reducerea timpului de căutare într-un tablou ordonat, se poate folosi o metodă mult mai eficientă: intervalul în care se caută valoarea este redus pas cu pas până la găsirea acesteia sau până când domeniul nu mai poate fi redus (valoare inexistentă). Principiul metodei este foarte bine pus în evidență de căutarea unui abonat în cartea de telefoane: se deschide cartea la jumătate; dacă abonatul căutat se află chiar pe poziția jumătății, căutarea se încheie cu succes; altfel, în funcție de numele abonatului (poziția în alfabet a primei litere din numele său) se continuă căutarea, doar în jumătatea din stânga sau în jumătatea din dreapta, după cum litera se află în alfabet înainte sau după litera de pe poziția jumătății.

Algoritmul metodei

Se determină poziția de mijloc, $m = \lfloor (1+n)/2 \rfloor$ (unde n reprezintă numărul de elemente) și se compară valoarea căutată cu $v[m]$:

– dacă valoarea căutată este mai mică decât a elementului din mijloc - $v[m]$ atunci se va căuta valoarea v_m printre elementele situate în stânga acestuia;

– dacă valoarea căutată este mai mare decât a elementului din mijloc - $v[m]$ atunci se va căuta valoarea v_m printre elementele situate în dreapta acestuia;

– altfel, valoarea v_m este egală cu elementul din mijloc - $v[m]$ atunci căutarea se încheie cu succes (valoare găsită pe poziția m din tablou).

Exemplu numeric:

Viteza vântului [km/oră] în prima săptămână a lunii mai 2006							
n=7 (o săptămână)							
v	12.34	15.90	18.34	24.45	34.58	38.95	45.25
Dacă $v_m = 38.95$							
Pas 1	$m = (1+7)/2 = 4$			$v_m > v_m$			
Se continuă căutarea la dreapta lui $v[m]$							
v	12.34	15.90	18.34	24.45	34.58	38.95	45.25
Pas 2				$m = (5+7)/2 = 6$		$v_m = v_m$	
Căutarea se încheie cu succes - valoarea căutată a fost găsită pe poziția 6							

3. Modificați algoritmul de căutare binară pentru cazul în care valoarea căutată este mai mare decât orice element din șir sau în cazul în care valoarea căutată este mai mică decât orice element din șir.

4. După susținerea probelor pentru examenul de admitere la o facultate, candidații obțin un anumit punctaj. Candidații sunt ordonați descrescător după medie. Să se scrie un algoritm care să afișeze numele și poziția în clasamentul final al unui candidat care a obținut un punctaj p .

6.5. Actualizarea grupurilor de date — editarea datelor

Actualizarea sau editarea datelor înseamnă modificarea, adăugarea sau ștergerea de informații dintr-o colecție de date.

STUDIU DE CAZ

Alexandra și Răzvan au deschis o firmă de design vestimentar. Pentru început, firma are n angajați care lucrează în departamentele firmei: informatizare, administrație, publicitate, creație. Fiecare angajat este identificat prin:

- cod (unic pentru fiecare angajat)
- nume
- prenume
- data nașterii
- cod departament
- funcție
- data angajării
- data încetării contractului
- salariu
- primă.

Informaticianul firmei introduce datele pentru fiecare angajat și completează un grup de date despre **angajați**. Acest grup este un tablou de înregistrări (articole). În Tabelul 6 este prezentată structura articolului și exemple de instanțe (valori pentru elementele grupului).

Tabelul 6

Nr. crt.	Cod	Nume	Prenume	Data nașterii	Cod depart	Data angaj.	Data plecării	Salariu RON	Prima %
1	345	Simion	Cătălina	15.06.76	IT	20.06.2005		1200	-
2	123	Zamfir	Andrei	21.01.80	PERS	01.06.2003		1500	15
3	452	Lazăr	Petru	14.11.78	DESIGN	12.03.2004		1900	-
4	561	Danciu	Ovidiu	07.08.82	ADVERT	01.06.2005		1600	20
5	190	Nasta	Ana	11.10.79	PERS	15.04.2002		1800	-

În timp, firma se dezvoltă și obține profit. Managerii decid ca firma să se extindă prin înființarea unor noi departamente, ceea ce atrage angajarea de personal. Întrucât firma obține profit, salariul angajaților crește; angajații care s-au evidențiat primesc prime, iar alții sunt promovați în funcții superioare.

Dacă un angajat se decide să părăsească firma sau dacă managerii firmei consideră că un angajat nu mai este util firmei și îl anunță că este concediat, atunci angajatul respectiv trebuie să fie exclus (șters) din grupul de date.

Managerii pot solicita în orice moment anumite situații:

- toți angajații care lucrează într-un anumit departament,
- persoanele care au fost angajate în anul înființării firmei sau în anul curent,
- angajații care sunt născuți în luna curentă,
- angajații care au avut primă luna trecută.

Aceste modificări impuse de strategia firmei trebuie să se regăsească și în valorile grupului de date; informaticianul firmei *actualizează-editează datele*.

6.5.1. Vizualizarea și tipărirea datelor

Vizualizarea și tipărirea datelor permit accesul la:

- toate componentele din colecția de date și la toate valorile acestora;
- toate componentele din colecția de date și la anumite valori ale acestora;
- toate componentele din colecția de date care îndeplinesc anumite condiții.

Prin vizualizare și tipărire, conținutul colecției de date nu se modifică.

Exemplu

Managerii firmei doresc să cunoască toți angajații (nume, prenume, salariu) din departamentul Design; pentru rezolvarea acestei cerințe, se tipărește următorul tabel:

Angajații din departamentul DESIGN

Nr. crt.	Nume	Prenume	Cod depart	Salariu RON
1	Nasta	Ana	DESIGN	1800

6.5.2. Modificarea datelor

Prin *modificare*, valoarea (informația) unei caracteristici se schimbă.

Exemple

1. Angajata Nasta Ana se mută de la departamentul PERS la departamentul DESIGN; valoarea câmpului *Cod depart* se *modifică*:

Nr. crt.	Cod	Nume	Prenume	Data nașterii	Cod depart	Data angaj.	Data plecării	Salariu RON	Prima %
5	190	Nasta	Ana	11.10.79	DESIGN	15.04.2002		1800	-

2. Angajatul Lazăr Petru pleacă din localitate și trebuie să părăsească firma începând cu data de 1.05.2006. Valoarea câmpului *Data plecării* corespunzătoare angajatului Lazăr va fi completată cu data încheierii contractului. Se realizează tot o operație de *modificare*.

Nr. crt.	Cod	Nume	Prenume	Data nașterii	Cod depart	Data angaj.	Data Plecării	Salariu RON	Prima %
2	123	Zamfir	Andrei	21.01.80	PERS	01.06.2003		1500	15
→ 3	452	Lazăr	Petru	14.11.78	DESIGN	12.03.2004	01.05.2006	1900	-

6.5.3. Adăugarea datelor

Prin *adăugarea* unui element într-un grup de date se completează cu un set de valori toate caracteristicile (câmpurile) acestui nou element.

Noul element se introduce după ultimul element din grupul de date.

Se pot adăuga unul sau mai multe elemente. După fiecare adăugare, numărul de elemente din grup se mărește cu o unitate.

Exemplu

La o nouă angajare se completează în colecția *angajați* toate informațiile despre angajat: cod, nume, prenume, prenume, data nașterii, cod departament, data angajării, salariu.

Nr. crt.	Cod	Nume	Prenume	Data nașterii	Cod depart	Data angaj.	Data plecării	Salariu RON	Prima %
1	345	Simion	Cătălina	15.06.76	IT	20.06.2005		1200	-
2	123	Zamfir	Andrei	21.01.80	PERS	01.06.2003		1500	15
3	452	Lazăr	Petru	14.11.78	DESIGN	12.03.2004		1900	-
4	561	Danciu	Ovidiu	07.08.82	ADVERT	01.06.2005		1600	20
5	190	Nasta	Ana	11.10.79	DESIGN	15.04.2002		1800	-
6.	589	Lucaci	Florina	21.04.84	PERS	01.05.2006		1200	

6.5.4. Inserarea datelor

Prin *inserarea* unui element într-un grup de date se completează cu un set de valori toate caracteristicile (câmpurile) acestui nou element.

Noul element se introduce între două elemente ale grupului; înainte de a insera un nou element, trebuie să localizăm poziția în care acesta va fi inserat.

Se pot insera unul sau mai multe elemente. După fiecare inserare, numărul de elemente din grup se mărește cu o unitate.

Exemplu

Informațiile despre noul angajat Roșca Mihai se vor insera înaintea înregistrării corespunzătoare angajatului Lazăr. Noul angajat are poziția 3 în colecția de date, iar Lazăr are poziția 4.

Nr. crt.	Cod	Nume	Prenume	Data nașterii	Cod depart	Data angaj.	Data plecării	Salariu RON	Prima %
2	123	Zamfir	Andrei	21.01.80	PERS	01.06.2003		1500	15
→ 3	452	Lazăr	Petru	14.11.78	DESIGN	12.03.2004	01.05.2006	1900	-

Nr. crt.	Cod	Nume	Prenume	Data nașterii	Cod depart	Data angaj.	Data plecării	Salariu RON	Prima %
2	123	Zamfir	Andrei	21.01.80	PERS	01.06.2003		1500	15
→ 3	354	Roșca	Mihai	08.09.79	IT	15.04.2006			
4	452	Lazăr	Petru	14.11.78	DESIGN	12.03.2004	01.05.2006	1900	-

6.5.5. Ștergerea datelor

Prin *ștergerea* unui element într-un grup de date se elimină toate valorile care îl definesc. Pentru ștergere, elementul trebuie mai întâi localizat.

Se pot șterge unul sau mai multe elemente. După fiecare ștergere, numărul elementelor din grup se micșorează cu o unitate.

Exemplu

După demisia angajatului Lazăr Petru, toate informațiile despre acesta sunt eliminate din **grupul angajați**.

Nr. crt.	Cod	Nume	Prenume	Data nașterii	Cod depart	Data angaj.	Data plecării	Salariu RON	Prima %
3	354	Roșca	Mihai	08.09.79	IT	15.04.2006			
→ 4	452	Lazăr	Petru	14.11.78	DESIGN	12.03.2004	01.05.2006	1900	-
5	561	Danciu	Ovidiu	07.08.82	ADVERT	01.06.2005		1600	20

Nr. crt.	Cod	Nume	Prenume	Data nașterii	Cod depart	Data angaj.	Data plecării	Salariu RON	Prima %
3	354	Roșca	Mihai	08.09.79	IT	15.04.2006			
4	561	Danciu	Ovidiu	07.08.82	ADVERT	01.06.2005		1600	20

TEME

1. Realizați descrierea elementelor din grupul de date cărți.
2. Întocmiți o listă cu prelucrările ce pot fi aplicate grupului de date cărți.
3. Specificați împrejurările în care asupra grupului de date cărți sunt necesare următoarele prelucrări:
 - modificare;
 - ștergere;
 - ordonare (precizați criteriul).
4. Realizați descrierea elementelor din grupul de date cititori.
5. Întocmiți o listă cu prelucrările ce pot fi aplicate grupului de date cititori.
6. Specificați împrejurările în care asupra grupului de date cărți sunt necesare următoarele prelucrări:
 - adăugare;
 - modificare;
 - ștergere;
 - ordonare (precizați criteriul).

Teste

1. Selectați varianta de răspuns care descrie proprietatea de generalitate a unui algoritm:
- a) rezolvă cât mai multe probleme;
 - b) rezolvă toate problemele de același tip oferind date de ieșire corecte pentru date de intrare corecte;
 - c) prelucrează cât mai multe date de intrare;
 - d) generalizează datele problemei pentru a fi supuse unui număr cât mai mare de prelucrări.

2. Asociați datelor din coloana **A** tipurile corespunzătoare din coloana **B**:

A

B

- | | |
|-----------------------------------|-----------------------------|
| a) anul nașterii unui copil | a) numeric, întreg |
| b) numele unui copil | b) numeric, real |
| c) mediile elevilor dintr-o clasă | c) șir de caractere |
| d) înălțimea unui copil | d) numeric, natural |
| | e) vector cu valori întregi |
| | f) vector cu valori reale |
| | h) fișier text. |

3. Apreciați cu A sau F valoarea de adevăr a următoarelor afirmații:

- a) Elementele unui vector ordonat descrescător respectă relația de monotonie $V[i] > V[i + 1], \forall i = 1, n$.
- b) Ordonarea unui vector necesită operații de comparare și interschimb.
- c) Într-un vector ordonat crescător, diferența a două elemente consecutive (de adrese i și $i + 1$) este întotdeauna pozitivă.
- d) Pentru a determina dacă un vector este ordonat, este suficientă o singură parcurgere.
- e) Un vector de caractere nu poate fi ordonat.

4. Asociați enunțurilor de probleme din coloana **A** prelucrările corespunzătoare din coloana **B**:

A

B

- | | |
|---|-----------------------------|
| a) Se dorește afișarea cronologică a n evenimente. | a) căutare secvențială |
| b) Se dorește afișarea primului restanțier din lista abonaților telefonici. | b) interclasare |
| c) Se dorește identificarea rapidă a unei persoane dintr-o agendă telefonică. | c) ordonare crescătoare |
| d) Se dorește afișarea cronologică a evenimentelor din două liste. | d) căutare binară |
| | e) interschimbare |
| | f) ordonare descrescătoare. |