

Liste

12.1. Liste liniare simplu înlanțuite

1. Să se creeze o listă liniară simplu înlanțuită, fiecare nod conținând câte un număr întreg, și apoi:
 - a) afișează numărul de noduri din listă;
 - b) afișează minimul dintre cheile din listă;
 - c) afișează cheile de pe pozițiile impare (prima, a treia, a cincea, etc);
 - d) afișează cheia nodului cu numărul de ordine k ;
 - e) verifică dacă cheile listei sunt în ordine descrescătoare;
 - f) afișează numărul de apariții a unei valori date k .
2. Fișierul **NUMERE.TXT** conține pe fiecare linie câte un număr întreg. Să se creeze și apoi să se tipărească o listă liniară simplu înlanțuită în care se vor introduce numai valorile pare din fișierul dat.
Exemplu. Dacă fișierul conține numerele 2, 3, 6, 5, 10, 1, câte unul pe o linie atunci lista va conține numerele 2, 6, 10.
3. Se dau două liste liniare simplu înlanțuite, informația din fiecare nod reprezentând un număr real. Să se construiască alte două liste liniare simplu înlanțuite care vor conține elementele comune celor două liste, respectiv elementele care se găsesc în prima listă și nu se găsesc în a doua listă.
Exemplu. Dacă prima listă conține numerele 2, 5, 3, 10, 22, 4, iar a doua 1, 5, 8, 4, 10 atunci în final prima listă va conține numerele 5, 10, 4, iar a doua 2, 3, 22.

4. Se citește de la tastatură un număr întreg cu maxim opt cifre. Să se afișeze oglinditul numărului folosind o listă liniară simplu înlănțuită în nodurile căreia se memorează cifrele numărului.
5. Se dau două liste liniare simplu înlănțuite. Creați o singură listă cu nodurile din cele două liste date, preluând alternative nodurile din cele două liste. De exemplu, dacă prima listă conține în ordine valorile {1, 2, 3} iar a doua listă conține valorile {7, 13, 1} lista finală va conține {1, 7, 2, 13, 3, 1}. Dacă una dintre liste este mai scurtă (se termină mai repede) nodurile din cealaltă listă vor fi toate adăugate la sfârșitul listei finale. De exemplu dacă prima listă conține valorile {1, 2, 3, 4, 5} iar a doua listă conține valorile {17, 18, 19} lista finală va conține {1, 17, 2, 18, 3, 19, 4, 5}.
6. Se citesc din fișierul **IN.TXT** datele despre notele la teza de informatică a unei clase de elevi. Fișierul conține:
 - pe prima linie numărul **n** de elevi;
 - pe fiecare din următoarele **n** linii, media, apoi numele unui elev, separate prin spațiu.

Folosind o listă liniară simplu înlănțuită, se cere:

- a) să se afișeze numele elevului (elevilor) care au nota cea mai mare;
- b) să se determine media generală a clasei.

7. Se dă o mulțime de puncte în plan. Se cere:
 - ♦ să se creeze o listă simplu înlănțuită care conține coordonatele (**x, y**) ale punctelor date;
 - ♦ să se afișeze punctele din listă care se află în interiorul cercului $C(x_0, y_0, R)$, unde x_0, y_0, R se citesc de la tastatură.

Exemplu. Dacă nodurile date sunt (2, 1), (6, 4), (1, 5), (4, 3), (3, 3), (2, 2) iar $x_0=3, y_0=3, R=2$ atunci punctele care se vor afișa sunt: (2, 1), (4, 3), (3, 3), (2, 2).

8. Să se creeze o listă liniară simplu înlănțuită cu articole având structura:

| Nume_student | An | Grupa | Media |
|--------------|----|-------|-------|
|--------------|----|-------|-------|

Să se listeze toți studenții cu **Media**>9, apoi să se elimine din listă toți studenții din anul 5.

9. Se dă o listă liniară simplu înlănțuită. Să se scrie o procedură care elimină elementele de pe pozițiile numere prime.

Exemplu. Dacă lista conține numerele 2, 9, 13, 5, 67, 11, 19, 18 atunci în final lista va fi 2, 5, 11, 18.

10. Se dă o listă liniară simplu înlănțuită. Să se scrie o procedură care adaugă după fiecare element de pe pozițiile impare p , un număr de p elemente având ca și conținut valoarea elementului de pe poziția p .

Exemplu. Dacă lista inițială este 2, 4, 6, 11, după adăugare ea va conține 2, 2, 4, 6, 6, 6, 6, 11.

11. Se dă șirul x cu n numere naturale citite de la tastatură. Să se creeze o listă liniară simplu înlănțuită conținând aceste numere. Să se elimine din listă elementele x_i pentru care suma elementelor x_1, \dots, x_{i-1} este egală cu x_i . Să se elimine apoi din listă elementele negative.
12. Un centru de închiriat casete video întocmește o listă (simplu înlănțuită) a casetelor disponibile pentru închiriere. Pentru fiecare casetă vor fi trecute în listă titlul filmului, numele regizorului și durata în minute. Scrieți un program care în funcție de dorința utilizatorului realizează următoarele acțiuni:
- adăugarea datelor aferente unei noi casete în listă;
 - ștergerea din listă a unei casete cu titlul dat de la tastatură;
 - afișarea întregii liste;
 - afișarea casetelor cu filmele regizate de un regizor cu nume introdus de la tastatură;
 - căutarea în listă a unui anumit film cu titlu dat de la tastatură.
13. Să se scrie un subprogram care elimină unul sau două (dacă lista conține un număr par de noduri) elemente din mijlocul unei liste liniare simplu înlănțuite. Subprogramul primește ca parametru adresa primului element al listei.

Exemplu. Dacă lista inițială este 2->5->19->12->33 atunci în final ea va conține 2->5->12->33.

14. Fie dată o listă liniară având nodurile de forma:

| | |
|-------|-----|
| cheie | urm |
|-------|-----|

unde **cheie** este un număr întreg, spre a cărei prim nod pointează variabila pointer **FIRST**. Lista este ordonată crescător după câmpul cheie astfel încât primul și ultimul nod conțin cea mai mică, respectiv cea mai mare valoare. Se cere să se scrie un program care șterge din listă un număr de noduri consecutive pentru care valoarea cheie este mai mare sau egală decât **KMIN** și mai mică sau egală decât **KMAX**.

Exemplu. Pentru **KMIN=25**, **KMAX=40** și lista 10->15->25->29->37->40->50, va rezulta lista 10->15->50.

15. Se dă o listă simplu înlănțuită care conține coordoatele (x, y) ale unor puncte din plan. Se cere:

♦ Să se listeze punctele care se află pe prima bisectoare;

- ♦ Să se elimine punctele aflate pe axa Ox sau Oy și apoi să se afișeze punctele rămase în listă;
- ♦ Să se insereze înaintea punctelor care nu se află pe prima bisectoare, punctul simetric față de aceasta;

Exemplu. Pentru lista $(19, 3), (3, 3), (-5, 0), (4, 9), (10, 10), (0, 10)$

- a) se listează simplu $(3, 3), (10, 10)$
 - b) se elimină punctele $(-5, 0), (0, 10)$ deci lista rămasă va fi $(19, 3), (3, 3), (4, 9), (10, 10)$.
 - c) lista devine $(3, 19), (19, 3), (3, 3), (9, 4), (4, 9), (10, 10)$.
16. Pentru organizarea unei selecții pentru un rolul principal feminin al unui film este necesară alcătuirea unei liste dinamice simplu înlănțuită, în care fiecare nod conține patru câmpuri de informație:
- numele și prenumele actriței
 - culoarea părului
 - culoarea ochilor
 - nota acordată de comisia de selecție (formată exclusiv din bărbați ☺)
- a) Să se creeze lista cu toate actrițele care participă la selecție
 - b) Să se listeze primele n concurente în ordinea descrescătoare a notelor.
 - c) Din lista de la punctul a) să se șteargă fizic toate concurente blonde cu ochii verzi și să se afișeze apoi concurente rămase.
17. Presupuneți că aveți o listă liniară simplu înlănțuită spre începutul căreia pointează variabila pointer **START**. Fiecare nod conține o cheie de căutare **cheie**, un câmp de informații **info**, și un pointer **urm** spre următorul nod. Să se scrie o funcție **CAUT(x, DATA)** care simultan caută și reorganizează lista în felul următor. Un nod cu cheia de căutare x este căutat. Nodurile listei sunt examinate secvențial. Dacă nodul este găsit, el este șters din poziția sa curentă și este mutat la începutul listei. Informațiile conținute în câmpul **info** al nodului sunt returnate în parametrul **DATA**. Valoarea returnată de funcție este **TRUE** dacă elementul cerut a fost găsit și **FALSE** în caz contrar.
18. Scrieți o procedură recursivă care inserează o valoare într-o listă simplu înlănțuită ordonată. Parametrii procedurii sunt: un pointer spre primul nod al listei simplu înlănțuite și valoarea de inserat. *Observație.* Procedura recursivă nu va conține nici o instrucțiune repetitivă.
19. Se dă o mulțime de articole de forma:

(Nr_legitimatie, Nume_student, Nota).

Se cere:

- ♦ să se creeze o listă simplu înlănțuită care conține aceste articole;
- ♦ să se ordoneze articolele descrescător după **Nota**, prin modificarea legăturilor dintre articole în cazul în care este nevoie de intershimbare;
- ♦ să se listeze articolele din lista sortată.

20. Se consideră un fișier text cu numele dat de la tastatură, ce conține elementele unui șir x_1, x_2, \dots, x_n ($n \geq 1$, x_i de tip byte). Se cere:

- 1) să se memoreze șirul într-o listă liniară simplu înlănțuită;
- 2) să se listeze șirul memorat;
- 3) să se determine un subșir de lungime maximă, cu elementele ordonate crescător (ordinea aparițiilor în șirul inițial se păstrează și în subșir);
- 4) să se determine toate subșirurile de lungime maximă cu proprietatea de la punctul anterior.

Exemplu. Pentru șirul inițial: 1, 7, 2, 3, 9, 4, 5, 12, subșirul crescător de lungime maximă este 1, 2, 3, 4, 5, 12.

21. Se dă o listă simplu înlănțuită în care fiecare nod reține în câmpul **INFO** un număr real iar în câmpul **urm** adresa următorului nod al listei. Să se scrie câte o procedură care tipărește elementele listei (**INFO**), de la cap la coadă și invers fără a crea alte liste, fără a modifica lista dată și printr-o singură parcurgere de fiecare dată.

22. Scrieți o procedură recursivă care primește o valoare de tip caracter și un pointer spre primul element al unei liste având același tip ca și în problema precedentă. Procedura va afișa codurile corespunzătoare fiecărei apariții a caracterului dat în ordinea inversă apariției lui în listă.

Exemplu. Pentru lista ('c', 1) -> ('o', 56) -> ('c', 23) -> ('a', 99) și caracterul 'c' se vor afișa valorile 23, 1.

23. Să se scrie o funcție care va inversa o listă simplu înlănțuită. Presupunem că fiecare nod conține în câmpul de informație un caracter.

Parametrul funcției va fi un pointer către primul nod al listei inițiale. Funcția va returna pointerul spre primul nod al listei inversate:

Exemplu Lista inițială 'A' -> 'B' -> 'C' va deveni 'C' -> 'B' -> 'A'.

24. Scrieți o procedură recursivă care primește o valoare de tip caracter și un pointer spre primul element al unei liste în care fiecare nod reține în câmpul **c** un caracter, în câmpul **cod** un număr întreg reprezentând o codificare a caracterului **c** iar în câmpul **urm** adresa următorului nod al listei. Procedura va afișa codurile corespunzătoare fiecărei apariții a caracterului dat în ordinea în care apare în listă.

Exemplu. Pentru lista ('c', 1) -> ('o', 56) -> ('c', 23) -> ('a', 99) și caracterul 'c' se vor afișa valorile 1, 23.

25. Se dau trei matrice rare (matrice cu un număr foarte mare de zero-uri), **A**, **B** și **C** de aceeași dimensiune $N \times N$, fiecare dintre ele reprezentată cu ajutorul câte unei liste liniare simplu înlanțuite. O listă va conține în nodurile sale valorile diferite de zero din matrice, împreună cu linia și coloana corespunzătoare. Să se construiască lista care va reprezenta matricea **D**, unde $D = A + B * C$.

Exemplu. Pentru matricele $A = \begin{pmatrix} 1 & 0 & 2 \\ 2 & 0 & 0 \\ 0 & 5 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 6 & 0 \\ 0 & 5 & 1 \\ 0 & 0 & 10 \end{pmatrix}$,

$C = \begin{pmatrix} 0 & 8 & 0 \\ 0 & 0 & 0 \\ 15 & 0 & 1 \end{pmatrix}$ cele trei liste vor fi:

A: (1, 1, 1) -> (2, 1, 3) -> (2, 2, 1) -> (5, 3, 2);

B: (1, 1, 1) -> (6, 1, 2) -> (5, 2, 2) -> (1, 2, 3) -> (10, 3, 3);

C: (8, 1, 2) -> (10, 3, 1) -> (1, 3, 3),

iar lista finală: (1, 1, 1) -> (8, 1, 2) -> (2, 1, 3) -> (17, 2, 1) -> (1, 2, 3) -> (150, 3, 1) -> (5, 3, 2) -> (10, 3, 3)

corespunzând matricei $\begin{pmatrix} 1 & 8 & 2 \\ 17 & 0 & 1 \\ 150 & 5 & 10 \end{pmatrix}$.

26. Se dau două mulțimi **A** și **B** a căror elemente, numere întregi, se memorează în două liste liniare simplu înlanțuite. Se cere să se scrie câte un program care calculează:

a) $C = A \cup B$

b) $C = A \cap B$

c) $C = A - B$

d) $C = (A - B) \cup (B - A)$

Se vor considera pentru fiecare problemă două situații: 1) listele sunt ordonate crescător; 2) listele nu sunt ordonate.

27. Considerăm următoarea reprezentare a termenilor unui polinom de 3 variabile **x**, **y**, **z**, folosind o listă simplu înlanțuită cu nodurile de forma:

| | | | | |
|---------------|---------------|---------------|-------|-----|
| puterea lui x | puterea lui y | puterea lui z | coef. | urm |
|---------------|---------------|---------------|-------|-----|

Presupunem că lista este ordonată după următoarea regulă: termenul $D_1 x^{A_1} y^{B_1} z^{C_1}$ va precede termenul $D_2 x^{A_2} y^{B_2} z^{C_2}$ dacă $A_1 > A_2$ sau, dacă

A -urile sunt egale și $B_1 > B_2$, sau dacă A -urile și B -urile sunt egale și $C_1 > C_2$.

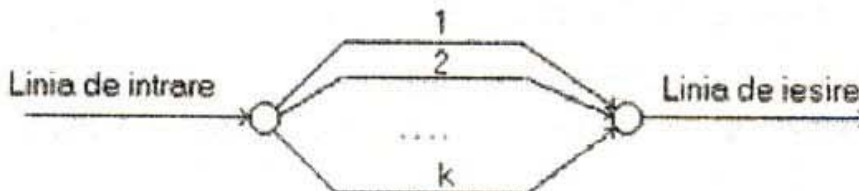
- a) Să se implementeze o funcție care inserează un nod într-o astfel de listă.
- b) Scrieți un algoritm care creează o listă liniară ordonată ca mai sus și care nu conține termeni duplicați.
- c) Fiind date două polinoame a căror prim nod este referit de pointerii P și respectiv Q , să se scrie o funcție care determină suma celor două polinoame și returnează pointerul spre lista reprezentând această sumă.
- d) Fiind date două polinoame a căror prim nod este referit de pointerii P și respectiv Q , să se scrie o funcție care determină diferența celor două polinoame și returnează pointerul spre lista reprezentând această diferență.
- e) Fiind date două polinoame a căror prim nod este referit de pointerii P și respectiv Q , să se scrie o funcție care determină produsul celor două polinoame și returnează pointerul spre lista reprezentând acest produs.
- f) Fiind dat un polinom a căror prim nod este referit de pointerul P , să se scrie un algoritm care parcurge lista și calculează valoarea polinomului pentru punctul (x, y, z) citit de la tastatură.
- g) Scrieți un algoritm de împărțire a polinomului P la polinomul Q . Polinomul cât va fi plasat într-o listă liniară simplu înlănțuită a cărei prim nod este dat de variabila pointer C , în timp ce restul este memorat într-o listă înlănțuită a cărei prim nod are adresa memorată în R .
- h) Scrieți un algoritm care integrează un polinom de 3 variabile în funcție de variabila x . Rezultatul va fi memorat într-o listă liniară simplu înlănțuită.

28. Se dau două polinoame $P(x)$ și $Q(x)$ de grad n , respectiv m , reprezentate ambele sub forma unei liste liniare simplu înlănțuită, fiecare nod conținând un coeficient al polinomului împreună cu puterea corespunzătoare a lui x . Să se calculeze: $P(x) + Q(x)$, $P(x) * Q(x)$, $P(Q(x))$ rezultatul memorându-se de asemenea în câte o listă liniară simplu înlănțuită.

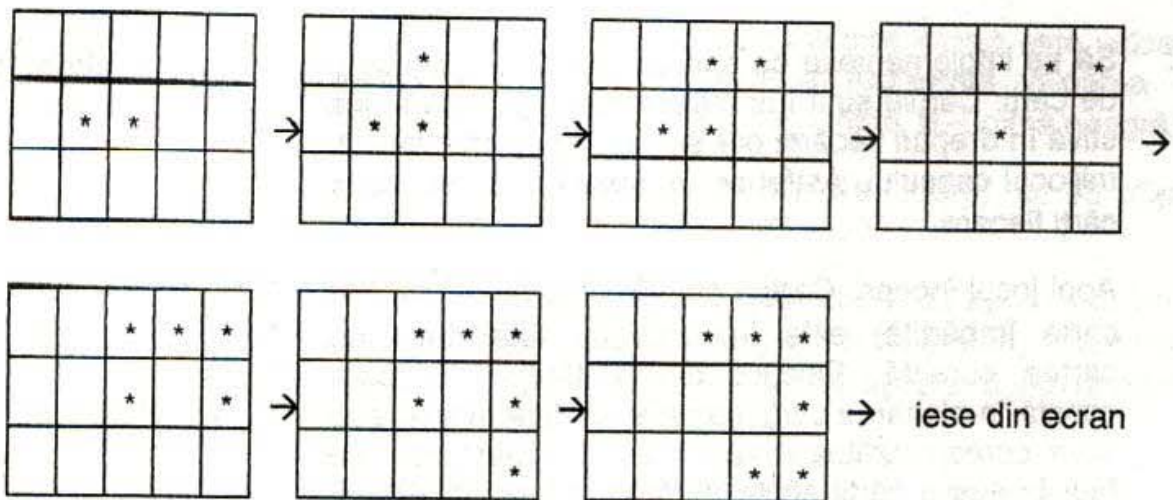
29. Se consideră un depou de locomotive cu o singură intrare și cu o singură linie ferată, care poate cuprinde oricâte locomotive. Să se scrie un program care realizează dispecerarea locomotivelor din depou. Programul prelucrează comenzi de forma: I (intrarea unei locomotive), E (ieșirea unei locomotive), L (listarea locomotivelor din depou) și S (sfârșitul programului). La terminarea programului trebuie să se afișeze locomotivele existente în depou.

Locomotivele pot intra și ieși din depou numai prin intrarea depoului, determinând în program o structură dinamică de tip stivă.

30. Aceeași problemă cu diferența că depoul are intrarea la un capăt și ieșirea la capătul opus. Locomotivele intră în depou pe la intrare și ies pe la ieșire, în aceeași ordine în care au intrat.
31. Pe o linie de cale ferată se găsesc, într-o ordine oarecare, N vagoane numerotate de la 1 la N . Linia se continuă cu alte k linii de manevre, ca în figura de mai jos. Cunoscând ordinea inițială a vagoanelor, să se obțină la ieșire vagoanele în ordinea $1, 2, \dots, N$; o linie este suficient de lungă astfel încât să încapă pe ea toate vagoanele.



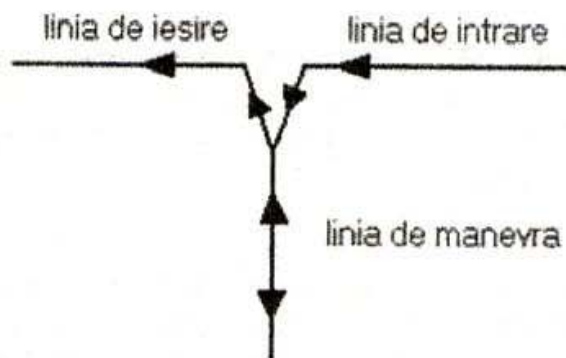
32. Pentru triajul din figura anterioară având la intrare vagoanele $1, 2, \dots, n$ într-o ordine oarecare să se afișeze toate posibilitățile lor de obținere pe linia de ieșire.
33. Se citește un șir de litere mici ale alfabetului englez, terminat cu caracterul '\$'. Se cere să se pună într-o stivă alocată dinamic toate literele citite. Se citește, apoi, o literă y din alfabet. Să se genereze două structuri dinamice de tip stivă, prima cuprinzând literele din stiva inițială care preced în alfabet litera y , cealaltă cuprinzând literele din stiva inițială care succed litera y .
34. Să se simuleze, folosind structuri dinamice de date, jocul "gâscă roșie" jucat de doi jucători. Un pachet de 32 de cărți este împărțit în mod egal celor doi jucători. Cărțile se țin cu fața în jos, jucătorii pun alternativ pe masă cu fața în sus cartea care se află deasupra pachetului lor. Dacă s-a pus o carte roșie, celălalt jucător trebuie să ia întregul pachet de cărți de pe masă și-l pune sub pachetul său. Jocul continuă în acest fel până când unul din jucători rămâne fără cărți. Acesta câștigă jocul.
35. Se consideră un ecran de dimensiune $m \times n$ ($m \leq 25$, $n \leq 80$) și o secvență de poziții ale elementelor situate pe ecran având proprietatea că oricare două elemente succesive aflate pe aceeași linie sau coloană (fără spații între ele) sunt vecine. Elementele secvenței sunt marcate cu '*'. Se citesc pozițiile secvenței, o valoare $k \in \mathbb{N}$ și un șir de comenzi. Șirul poate să conțină comenzile 'N', 'S', 'E', 'V', al căror efect este adăugarea unui nou element '*' în poziția corespunzătoare comenzii, față de ultimul element al secvenței. Din k în k comenzi, primul element al secvenței dispare. Se cere:
- validarea datelor de intrare (a secvenței de poziții și a șirului de comenzi);
 - să se semnaleze momentul când în urma unei comenzi se ajunge într-o poziție existentă sau când în urma unei comenzi secvența ar ieși din ecran.
- Exemplu.* Pentru $m=3$, $n=5$, $k=3$, secvența de poziții $(2, 2)$, $(2, 3)$, șirul de comenzi 'NESSVS' se va semnala ieșirea din ecran:



Observații. Pentru rezolvare nu se vor folosi vectori sau matrice, ci doar liste simplu înlanțuite.

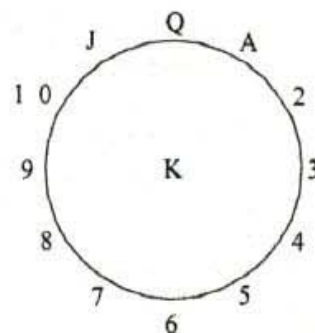
36. Într-un triaj există o linie de cale ferată pentru manevre, ca în figura de mai jos, pe linia de intrare sunt n vagoane: $1, 2, \dots, n$. Pe linia de intrare și de ieșire deplasarea se poate face numai în sensul indicat de săgeți; pe linia de manevră deplasarea se poate face în ambele sensuri.

- cunoscând ordinea vagoanelor la intrare și ordinea în care se dorește să apară ele la ieșire, să se afișeze, dacă problema are soluție, manevrele necesare.
- aceeași problemă ca la punctul a), în condițiile în care și pe linia de intrare este permisă deplasarea în ambele sensuri;



37. Pe o tijă verticală sunt n bile colorate cu cel mult k culori, fiecare bilă având o etichetă cu un număr de la 1 la n . Să se mute bilele pe alte k tije, pe fiecare punând numai bile de aceeași culoare. Fiecare tijă are un capăt liber pe unde se introduc sau extrag bilele, celălalt capăt fiind sudat pe suportul tijei. Să se afișeze bilele de pe fiecare din cele k tije (se vor utiliza structuri dinamice).

38. Să se implementeze cu ajutorul alocării dinamice a memoriei următorul joc de cărți: Cărțile sunt împărțite în cerc, cu fața în jos, formând un ceas, cu o stivă în dreptul fiecărei ore și o stivă suplimentară în mijlocul ceasului. Astfel se formează 13 stive, cu 4 cărți fiecare.



Apoi jocul începe. Cartea din vârful stivei "K" (ultima carte împărțită) este întoarsă cu fața devenind cartea curentă. Fiecare din mutările următoare constă în plasarea cărții curente, cu fața în sus, sub stiva corespunzătoare valorii ei, și întoarcerea, cu fața în sus a cărții aflate în vârful stivei respective.

Aceasta va deveni cartea curentă. De exemplu dacă un "As" este cartea curentă, ea va fi pusă sub stiva de la ora 1, și cartea din vârful stivei devine cartea curentă. Jocul se termină când stiva indicată de cartea curentă nu mai conține nici o carte cu fața în jos.

Intrarea. Constă din patru linii conținând fiecare câte 13 cărți, separate între ele printr-un spațiu. Fiecare carte este reprezentată prin două caractere, prima este valoarea cărții (A, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, unde T reprezintă valoarea 10) urmată de tipul cărții (I – inimă roșie, R – romb, T – treflă, N – inimă neagră). Cărțile sunt listate de jos în sus, adică prima carte împărțită este ultima carte din fișierul de intrare.

Ieșirea. Constă dintr-o singură linie, conținând numărul de cărți cu fața în sus existente la sfârșitul jocului, urmat de ultima carte care a fost întoarsă, în formatul dat la intrare.

39. Scrieți un program care simulează funcționarea unei cozi. În mod repetat, cât timp utilizatorul dorește acest lucru, programul va permite alegerea uneia din operațiile aplicabile cozii: adăugarea unui element în coadă, eliminarea unui element, afișarea cozii.
40. Această problemă vă cere să simulați un alt joc de cărți. Cărțile sunt puse pe masă una câte una într-o linie de la stânga la dreapta, fără a le suprapune. Când o carte se potrivește cu cartea aflată în vârful stivei din stânga sa sau cu cartea din a treia stivă din stânga sa, se va muta în vârful stivei respective. Spunem că două cărți se potrivesc dacă ele au aceeași valoare sau sunt de același tip. Doar cartea din topul unei stive se poate muta. Spațiul care se poate crea între stive se va elimina prin mutarea tuturor stivelor din dreapta spațiului creat cu o poziție spre stânga. Jocul continuă până când toate cărțile sunt puse pe masă.

În cazul în care mai multe cărți pot fi mutate, se va muta cea mai din stânga carte posibilă. Dacă o carte se poate muta atât cu o poziție cât și cu trei poziții, ea se va muta trei poziții.

Intrarea. Intrarea constă din câte patru linii, fiecare linie conținând 13 de cărți separate printr-un spațiu. Linia finală a fișierului de intrare conține caracterul #.

Cărțile sunt este reprezentate prin câte două caractere, prima este valoarea cărții (A, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, unde T reprezintă valoarea 10) urmată de tipul cărții (I – inimă roșie, R – romb, T – treflă, N – inimă neagră).

Ieșirea. Pentru fiecare pachet de cărți (patru linii de intrare), se va afișa o linie, indicând numărul de cărți rămase în fiecare stivă.

41. Într-un super-market sunt n case de marcat, numerotate $1, 2, \dots, n$. Scrieți un program care simulează acțiunile a m cumpărători, numerotați $1, 2, \dots, m$, știind că în magazin lucrurile se desfășoară după următorul algoritm:

- din t în t secunde, câte un cumpărător dorește să plătească produsele achiziționate în regim de auto-servire, și pentru aceasta se așează la coadă la casa unde sunt cei mai puțini cumpărători: ordinea în care cumpărătorii se așează la coadă este cea naturală, dată de numerotarea lor;
- pentru fiecare cumpărător se cunoaște timpul necesar plății produselor achiziționate, exprimate în secunde (care depinde evident de numărul produselor cumpărate);
- după ce a plătit, fiecare cumpărător părăsește coada de așteptare.

Datele de intrare se citesc din fișierul 'MAGAZIN.IN', care conține:

- pe primul rând valorile m , n și t , separate prin spații;
- pe fiecare din următoarele m rânduri câte un întreg, reprezentând timpul necesar unui cumpărător pentru achitarea produselor achiziționate.

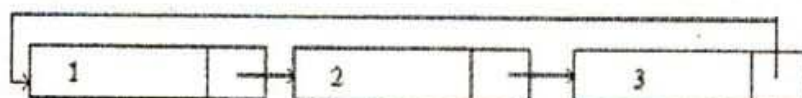
Programul va afișa "starea caselor de marcat" după fiecare intrare sau ieșire a unui cumpărător. Rezultatele se scriu în fișierul 'CASE.TXT', în care o intrare/ieșire a unui cumpărător este înregistrată pe două rânduri astfel:

- pe primul rând se scriu trei valori: prima valoare va fi caracterul 'i' sau 'e' indicând dacă s-a produs o intrare sau o ieșire a unui cumpărător, a doua valoare reprezintă numărul casei unde s-a produs "evenimentul", iar a treia valoare desemnează numărul de ordine al cumpărătorului intrat/ieșit;
- pe al doilea rând se va reprezenta modul cum arată, după intrarea/ieșirea respectivă, coada de așteptare din fața casei de marcat la care a avut loc evenimentul (se vor scrie numerele de ordine ale cumpărătorilor aflați în coadă separate prin spații).

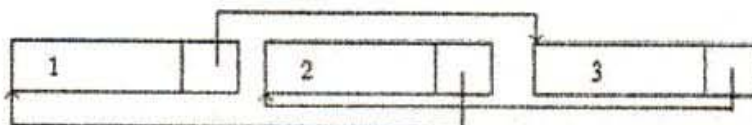
12.2. Liste circulare

42. Scrieți un algoritm de concatenare a două liste circulare simplu înlanțuite.
43. Se dă o listă circulară simplu înlanțuită, ale cărei noduri conțin coordonatele unor puncte în plan. Să se modifice referințele la elemente astfel încât noua listă să reprezinte vârfurile în ordine trigonometrică, a unei linii poligonale închise care trece prin toate punctele date o singură dată.
44. Elaborați un algoritm care să considere o listă circulară simplu înlanțuită și să inverseze direcțiile tuturor săgeților.

Exemplu. Din lista:



se va obține lista



45. Scrieți un algoritm de împărțire a unei liste circulare simplu înlanțuită în două liste circulare având un număr cât mai apropiat de elemente (dacă e posibil egal).
46. Pentru reprezentarea unui poligon convex se rețin în ordine coordonatele carteziane ale vârfurilor poligonului într-o listă înlanțuită circulară. Scrieți un program care să calculeze și să afișeze pe ecran aria unui poligon astfel reprezentat. Datele de intrare se citesc din fișierul text **POLIGON.TXT**, care are următoarea structură: pe prima linie se găsește numărul natural n al vârfurilor din poligon, următoarele n linii conțin perechi de câte două numere întregi reprezentând coordonatele în plan a celor n vârfuri ale poligonului.
47. Să se implementeze o variantă modificată a binecunoscutului joc "Nu te supăra frate". Tabla de joc este reprezentată de o listă circulară simplu înlanțuită, fiecare nod al listei conținând un număr natural pozitiv. Un singur nod conține valoarea zero. Acest nod este considerat ieșirea din joc. Ambii jucători pornesc din același nod. Un jucător aflat la mutare se deplasează cu nr căsuțe, unde nr este numărul memorat în nodul în care se află jucătorul. Valoarea din căsuța părăsită se modifică astfel încât noua valoare să fie egală cu partea întreagă a mediei aritmetice a valorilor memorate în cele două noduri vecine nodului respectiv. Jucătorii mută alternativ. Jocul se termină atunci când un jucător a ajuns la ieșire, acesta fiind declarat câștigător, sau dacă ambii jucători au efectuat m mutări fără a ajunge la ieșire. În acest caz este declarat câștigător jucătorul aflat cel mai aproape (în sensul legăturilor dintre nodurile listei) față de ieșire.

48. **Problema lui Josephus:** Un număr de n copii stau într-un cerc. La un moment dat, începând cu copilul m , se elimină din cerc al k -lea copil, cercul apoi strângându-se. Se cere să se afișeze ordinea în care sunt eliminați copiii din cerc, în funcție de n , m , k și de direcția de parcurgere a cercului.

Exemplu. Pentru $n=5$, $m=3$, $k=2$ și direcția stânga, se va obține ordinea: 5, 2, 1, 4.

12.3. Liste liniare dublu înlanțuite

49. Se consideră o matrice A de dimensiune $n \times m$, $n \leq 100$, $m \leq 100$. Să se reaseze elementele în matrice astfel încât ele să apară în ordine crescătoare atât pe linii cât și pe coloane.
50. Se citește un șir de cuvinte terminat cu '*'. Să se afișeze cuvintele în ordine inversă citirii și, apoi, în ordinea în care au fost citite, fără a utiliza șiruri.
51. Se dă o listă liniară dublu înlanțuită, informația din fiecare nod reprezentând o valoare numerică reală. Să se modifice referințele la elementele listei astfel încât noua înlanțuire să reprezinte ordonarea descrescătoare a elementelor listei.
52. Scrieți o funcție care construiește o listă liniară dublu înlanțuită dintr-o listă simplu înlanțuită care va fi accesată printr-un pointer **FIRST**.
53. Să se scrie un program care tratează comenzile de generare și prelucrare a unei liste dublu înlanțuite cuprinzând cuvinte. Comenzile afișate de o procedură meniu sunt:
- inserarea în fața unui cuvânt dat;
 - inserarea după un cuvânt dat;
 - ștergerea unui cuvânt dat;
 - traversarea listei de la început;
 - traversarea listei de la sfârșit;
 - oprirea programului.

În cazul în care cuvântul după care, sau înaintea căruia trebuie să se facă introducerea nu este găsit, elementul nou se adaugă la sfârșitul listei.

54. Să se scrie un subprogram care determină numărul de elemente dintr-o listă liniară dublu înlanțuită. Subprogramul va primi ca parametru adresa unui element oarecare al listei.

55. Se dă o listă liniară dublu înălțuită, având ca informație, în fiecare nod, o valoare numerică întreagă. Să se descompună această listă în trei liste astfel:
- prima să conțină elementele listei inițiale care dau 0 la împărțirea la 3;
 - a doua să conțină elementele care dau restul 1 la împărțirea cu 3;
 - și a treia listă conține restul elementelor din lista inițială.
- Exemplu.* Dacă lista inițială conține numerele 12, 9, 5, 7, 91, 3, 17, 22 atunci cele trei liste vor conține 12, 9, 3; a doua 7, 91, 22; respectiv 5, 17 ultima.
56. Se dă o listă liniară dublu înălțuită, informația din noduri fiind numere întregi. Să se modifice referințele la nodurile listei astfel încât să se obțină două liste liniare dublu înălțuite care vor conține elementele de pe pozițiile impare, respectiv pare din lista inițială.
- Exemplu.* Dacă lista inițială conține, pe rând, numerele 3, 7, 12, 9, 10, 21, 14 atunci prima listă rezultată va conține 3, 12, 10, 14, iar a doua 7, 9, 21.
57. Să se scrie un program care creează o listă liniară dublu înălțuită cu numere reale. Să se insereze apoi între oricare două noduri ale listei un nod nou conținând ca informație media aritmetică din cele două noduri.
- Exemplu.* Dacă în lista inițială sunt memorate numerele 2, 9, 12, 6, 1 atunci în final lista va conține 2, 5.5, 9, 10.5, 12, 9, 6, 3.5, 1.
58. Să se scrie un program care realizează următoarele cerințe. Opțiunile se vor include într-un meniu din care utilizatorul va avea posibilitatea să aleagă operația dorită.
- a) Să se creeze o listă liniară dublu înălțuită în care, fiecare nod, pe lângă informațiile de legătură, va conține date referitoare la un elev candidat la examenul de capacitate: **nume**, **prenume** și un vector cu 3 componente care rețin notele elevului la cele trei probe ale examenului (numere reale). Datele se vor citi de la tastatură.
 - b) Să se afișeze pe ecran **numele**, **prenumele** și **media** la capacitate a fiecărui elev din lista creată.
 - c) Să se calculeze media generală a tuturor candidaților care se găsesc în listă.
 - d) Să se memoreze informațiile din listă în fișierul text "**CAPAC.TXT**". Prima linie a fișierului va conține numărul candidaților. Fiecare din următoarele linii va conține informațiile referitoare la un elev.
 - e) Să se citească datele din fișierul text creat anterior și recreează lista.

- f) Să se adauge înregistrări noi la sfârșitul listei. Datele se citesc dintr-un fișier text care are aceeași structură ca la punctele anterioare și numele citit de la tastatură.
 - g) Să se inverseze două noduri consecutive din listă, noduri având numărul de ordine dat.
 - h) Folosind funcția sau procedura scrisă la punctul anterior să se sorteze descrescător elementele listei după media fiecărui elev.
 - i) Să se listeze toți elevii care au media mai mare sau egală cu o valoare citită de la tastatură.
 - j) Listați primii k elevi (în ordinea descrescătoare a mediilor). În cazul în care există mai mulți elevi cu aceeași medie pe poziția k se vor afișa toți.
 - k) Să se elimine din listă elevii cu numele și prenumele date de la tastatură.
59. Pentru a putea lucra cu numere întregi mari, care nu pot fi memorate în nici o variabilă de tip predefinit, se pot folosi listele liniare dublu înălțuite, fiecare nod al unei liste memorând o cifră a acestui număr. Folosind astfel de liste se cere să se calculeze:
- a) suma a două numere întregi mari;
 - b) scăderea a două numere întregi mari;
 - c) înmulțirea a două numere întregi mari;
 - d) împărțirea a două numere întregi mari;
 - e) compararea a două numere întregi mari;
 - f) maximul a n numere întregi mari;
 - g) factorialul unui număr întreg mare.

12.4. Alte structuri de tip listă

60. Într-un fișier text sunt păstrate numele unor noduri feroviare și ale punctelor finale ale rutelor ce pornesc din acel punct. Fișierul conține pe fiecare linie numele unui nod feroviar, urmat de ":", apoi numele localităților finale pentru fiecare ruta existentă din acel punct, separate prin spații. De exemplu:

Timisoara:Arad Jimbolia Buzias Resita Lugoj

Arad:Oradea Deva

Lugoj:Ilia Caransebes

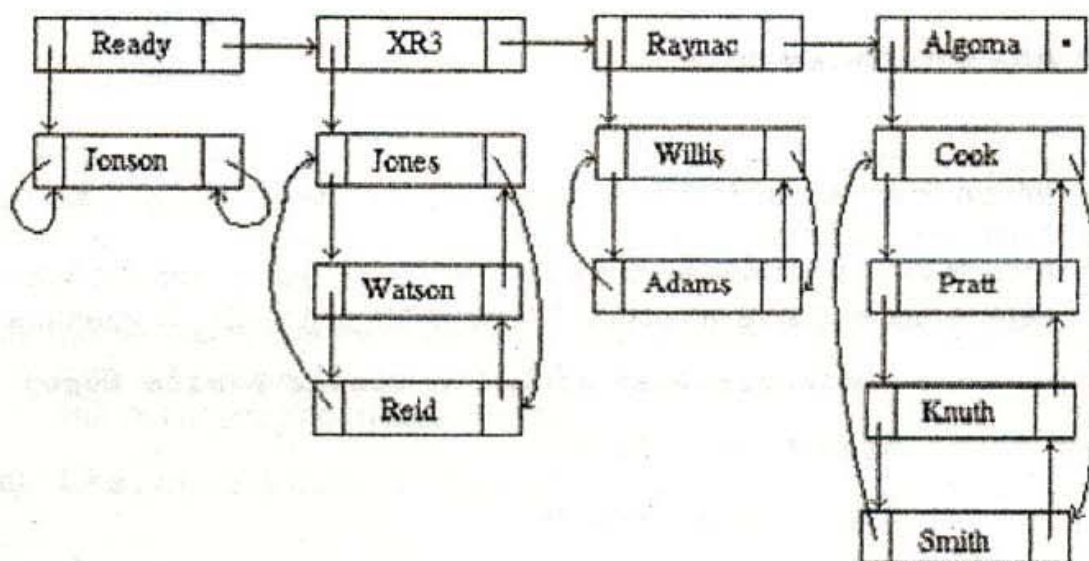
Deva:Cluj Alba Petrosani

Se cere:

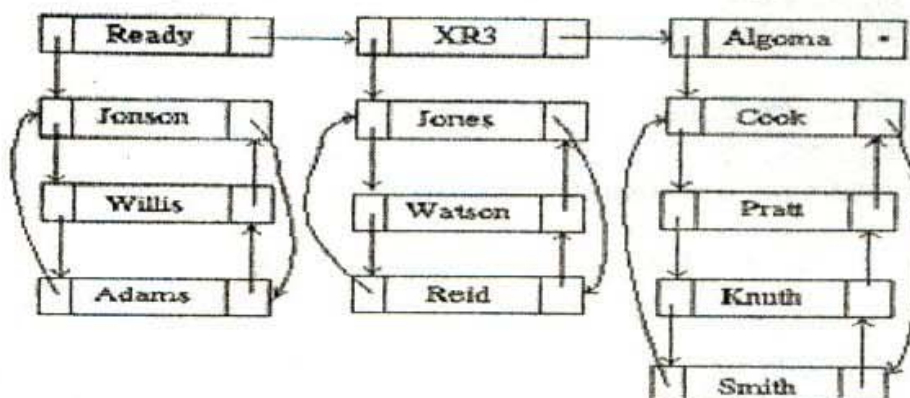
- Să se creeze o structură de tip listă care să păstreze informațiile din fișierul de intrare.
 - Verificați dacă un oraș, al cărui nume se citește de la tastatură, este sau nu nod feroviar.
 - Creați o lista care pastrează toate orașele extremități ale rutelor feroviare existente.
 - Verificați dacă între două orașe există legături directe. Numele orașelor se citesc de la tastatură.
 - Verificați dacă între două orașe există legături feroviare.
61. Un grup este format din $N+1$ indivizi: pentru a transmite diverse mesaje, șeful grupului A_0 folosește următorul procedeu de comunicare: A_0 comunică cu A_1 și A_2 , A_1 cu A_2 și A_3 , ..., A_{n-1} și A_n transmit mesajul de confirmare către A_0 . Dacă un individ părăsește grupul, legăturile se refac; dacă în grup apar indivizi noi, aceștia sunt plasați la sfârșitul lanțului de comunicare. Să se scrie un program care să simuleze acest sistem utilizând alocarea dinamică a datelor.
62. Pentru a ajuta la administrarea implementării unor proiecte software, un sistem de administrare a fost dezvoltat de firma **DataPro**. Prima sarcină a sistemului este să urmărească analiștii și programatorii la ce proiecte lucrează și care sunt disponibili pentru a lucra dar nu au fost încă asignați nici unui proiect.

Considerăm de exemplu structura de date din figura de mai jos. În figură, trei proiecte sunt curent active (**XR3**, **Raynac** și **Algoma**) și personalul pentru fiecare este listat dedesubtul numelui de cod al proiectului. Lista personalului disponibil pentru lucru este păstrat sub nodul proiect etichetat cu **Ready**.

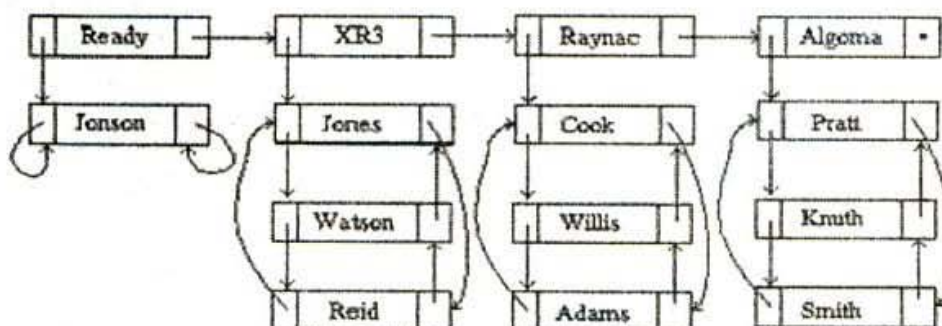
Presupunem o structură care conține câmpurile **LPTR**, **Name** și **RPTR** unde **Ready** este nodul pointat de **First**.



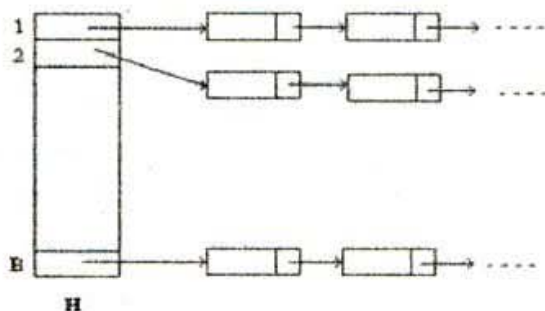
- a) Scrieți un algoritm care rezolvă cazul în care un proiect este terminat, toate persoanele asociate proiectului sunt atașate listei **Ready**. De exemplu structura de mai sus se modifică obținând figura următoare atunci când proiectul **Raynac** este terminat:



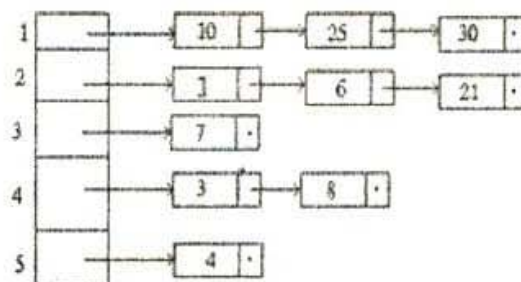
- b) Scrieți un algoritm care mută un programator de la un proiect la alt proiect. De exemplu, dacă **Cook** este mutat de la proiectul **Algoma** la proiectul **Raynac**, pentru structura inițială, atunci noua structură va arăta astfel ca în figura:



63. Pentru a memora un dicționar vom proceda după cum urmează: mulțimea membrilor potențiali ai dicționarului se împart într-un număr finit de clase. Dacă dorim să avem B clase, numerotate $1, 2, \dots, B$, atunci vom folosi o funcție h , numită funcție de **hash**, astfel încât pentru un obiect x având tipul membrilor mulțimii, $h(x)$ este unul din întregii $1, 2, \dots, B$. Valoarea $h(x)$ este clasa din care x va face parte. Fiecare clasă este organizată ca o listă liniară simplu înălțuită. Pointerii spre primul element al fiecărei liste se vor păstra într-un tablou H .



(a)



(b)

- a) Să se scrie o procedură care creează o astfel de structură, numită **open hashing**, în care se memorează numere întregi citite de la tastatură, știind că vom avea 5 clase iar funcția de hash va fi $h(x) = x \bmod 5 + 1$. În interiorul unei clase numerele vor fi păstrate în ordine crescătoare. De exemplu pentru șirul de intrare 1, 7, 10, 21, 4, 30, 8, 25, 6, 3, structura creată va arăta ca în figura (b).
- b) Fiind date două mulțimi **A** și **B** de numere întregi reprezentate folosind algoritmul de la punctul anterior să se scrie câte o procedură pentru fiecare din următoarele operații:
- 1) $A \cup B$ 2) $A \cap B$ 3) $A - B$.
- c) Să se scrie o procedură care pornind de la o structură open hashing de tipul celei de la punctul a), să creeze o listă liniară simplu înlănțuită care să conțină toate numerele din structură în ordine crescătoare.

Observație. Nu se vor folosi procedurile **new** și **dispose** ci se vor realiza doar modificări ale adreselor conținute în nodurile structurii. Primul nod al listei finale va fi indicat de pointerul **H[1]**.

64. Se dă o mulțime de mașini caracterizate prin **culoare**, **marcă**, **număr**. Să se construiască o listă cu culorile mașinilor date și, pentru fiecare culoare, o sublistă a mașinilor având acea culoare. Să se listeze numerele de înmatriculare ale mașinilor care au o culoare și o marcă dată.

Indicație. Se va folosi o structură recursivă ca cea din figura de mai jos:

