Lucrarea de Laborator nr. 5

Tema: Algoritmi de prelucrare a listelor liniare simplu înlănțuite, listelor dublu înlănțuite, listelor circulare, cozilor și stivelor

Scopul: Obținerea deprinderilor practice de implementare și de utilizare a tipurilor abstracte de date „Listă liniară simplu înlănțuită”, „Listă liniară dublu înlănțuită”, „Listă circulară”, „Stivă”, „Coadă” în limbajul C cu asigurarea operațiilor de prelucrare de bază ale listei.

Sarcina (pe variante)

1. Scrieți un program, care din lista *L1,* ce conține numere întregi, să extragă în lista *L2* numerele pare, în lista *L3* numerele impare, iar în lista *L4* cele nule.
2. Scrieți un program pentru a număra biții de la dreapta spre stânga, pentru fiecare număr introdus într-o coadă și le salvează în una nouă. Sa se scrie o funcție care șterge elementele din n în n (ca exemplu, dacă coada are 10 elemente, iar n=3, ștergem elementele cu numerele de ordine 3, 6 și 9).
3. Scrieți un program, care rotește fiecare element al listei dublu înlănțuite *n* la dreapta cu *b* poziții. Sa se scrie o funcție care, primind o lista, întoarce mulțimea tuturor perechilor din lista. Pentru [a,b,c,d] va produce [[a b],[a c],[a d],[b c],[b d],[c d]].
4. Scrieți un program de inversare a cei *n* biți ai elementelor unei liste simplu înlănțuită, care încep de la poziția *p*, lăsându-i pe ceilalți neschimbați și le salvează în una nouă. Sa se scrie o funcție care transforma o lista într-o mulțime (fără repetarea elementelor).
5. Scrieți un program, care sortează și convertește literele mici în litere mari pentru cuvintele dintr-o stivă și le salvează în una nouă. Sa se intercaleze o literă în listă ordonată, astfel încât lista rezultata să rămână ordonată.
6. Scrieți un program, care decide, dacă o valoare particulară x apare într-o listă dublu înlănțuită *v*. Elementele lui *v* trebuie să fie în ordine crescătoare. Se tipărește numărul elementului din listă (un număr între 0 și n-1), dacă *x* apare în *v***,** și –1, dacă nu. Sa se scrie o funcție care întoarce lista permutărilor unei liste date. Ex: pentru [1,2,3] furnizează L=[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]].
7. Scrieți un program, care convertește elementul unei liste ciclice (circulare) *n* în baza binară, într-un șir de caractere și le salvează în una nouă. Sa se scrie o funcție care șterge din lista elementele din n în n (ca exemplu, dacă lista are 10 elemente, iar n=3, ștergem elementele cu numerele de ordine 3, 6 și 9).
8. Scrieți un program, care convertește întregii fără semn dintr-o listă simplu înlănțuită *n***,** într-o reprezentare binară și le salvează în una nouă. Sa se scrie o funcție care întoarce lista submulțimilor unei liste date. Ex: pentru [1,2,3] furnizează [[],[1],[2],[3],[1,2],[1,3],[1,2,3]], nu neapărat în această ordine.
9. Scrieți un program, care convertește fiecare element al listei dublu înlănțuite într-un număr hexazecimal și le salvează în una nouă și să se înlocuiască toate aparițiile unui element E cu elementele unei alte liste, L1. Exemplu: inloc([1,2,1,3,1],1,[10,11],X) va produce X=[10,11,2,10,11,3,10,11].
10. Scrieți un program de convertire a fiecărui element dintr-o stivă într-un șir de caractere și invers, salvându-le în una nouă și sa se sorteze.
11. Scrieți un program, care inversează fiecare element de tip șir de caractere dintr-o listă simplu înlănțuită și le salvează în una nouă. Sa se sorteze lista cu păstrarea cuvintelor dubluri.
12. Scrieți un program, care calculează cel mai mare divizor comun al elementelor dintr-o coadă și le salvează într-o stivă nouă. Sa se scrie o funcție care adaugă din n în n un element dat: ex: în [1,2,3,4,5] pt. n=2 și element=6 => L=[1,2,6,3,4,6,5].

13. Scrieți un program, care compară două stive date și formează încă o stivă cu cele ce coincid. Sa se scrie o funcție care întoarce o lista cu pozițiile pe care apare elementul maxim într-o listă de numere întregi. Ex: pentru [1,-1,4,2,4] => L=[3,5].
14. Scrieți un program de calculare a numărului de elemente dintr-o listă simplu înlănțuită, care sunt mai mici ca valoarea medie aritmetică a tuturor elementelor acestei liste. Sa se scrie o funcție care testează egalitatea a două mulțimi.
15. Scrieți un program, care dintr-o listă circulară de 100 de numere aleatoare să determine numărul maximal și cel minimal. Să se determine consecutivitatea de elemente, ce se află între numărul maximal și cel minimal determinat. Să se scrie o funcție care testează inegalitatea a două mulțimi.
16. Scrieți un program, care din trei liste simplu înlănțuite să selecteze într-o listă nouă mai întâi numerele divizibile la 8, 4 și 2, apoi numerele negative impare. Se cere de a-l diviza în 3 subliste.
17. Scrieți un program, care atribuie unei liste simplu înlănțuite elementele altei liste în ordine inversă. Apoi de salvat în 2 fișiere, fiind ordonate descrescător, și se unesc în unul, păstrându-se ordinea descrescătoare de sortare.
18. Scrieți un program, care va tipări în ordine inversă subconsecutivitatea de numere dintre valoarea minimă și maximă ale unei liste simplu înlănțuită și se cere de a le diviza în 5 subliste.
19. Scrieți un program, care formează o listă dublu înlănțuită nouă din cea dată după următoarea legitate: elementele listei noi se obțin din inversul cifrelor numărului din lista dată. Definiți o funcție care șterge elementul minim din listă.
20. Scrieți un program cu funcții, care compară două stive date și calculează numărul de elemente din listă simplu înlănțuită, care sunt mai mici ca valoarea medie aritmetică a tuturor elementelor acestei liste.
21. Scrieți un program, care atribuie unei liste simplu înlănțuite elementele altei liste în ordine inversă.
22. Scrieți un program, care creează o listă circulară, a căror valori ale elementelor sunt cuprinse între 10 și 1000. Să se determine frecvența cu care a fost generat fiecare element al listei create.
23. Scrieți un program care determina numărul maximal și cel minimal într-o listată circulară de 100 de numere aleatoare. Sa se determine consecutivitatea de elemente ce se află între numerele maximal și minimal determinat.
24. Scrieți un program, care determină câte numere ale unei cozi de 100 de numere aleatoare sunt mai mari ca „vecinii” săi, salvându-le într-un fișier.
25. Alcătuiți un program, care ar efectua următoarele operații asupra listei dublu înlănțuite:
    - de determinare a primului element în listă;
    - de căutare a elementului după criteriul dat;
    - de inserare a unui element nou, înainte sau după o componentă indicată a listei;
    - de eliminare a unui element din listă;
    - de sortare a componentelor listei.
26. Scrieți un program, care ar efectua următoarele operații. Se creează o listă dintr-un șir de numere reale, care se termină cu zero, apoi din listă se șterg mai întâi elementele pozitive, apoi numerele impare. Sa se scrie o funcție care întoarce reuniunea a doua mulțimi de numere.
27. Scrieți un program de unire a patru cozi în una nouă, în care vor fi stocate mai întâi numerele negative, zerourile, apoi numerele pozitive.
28. Alcătuiți un program, care ar efectua următoarele operații asupra listei dublu înlănțuite:
    - de determinare a primului element în listă;
    - de căutare a elementului după criteriul dat;
    - de inserare a unui element nou, înainte sau după o componentă indicată a listei; de eliminare a unui element din listă. de sortare a componentelor listei.

29. Se dă o consecutivitate de numere întregi, care se termină cu 0. Să se alcătuiască un program pentru introducerea acestei consecutivități cu utilizarea metodei de stocare indexată consecutiv–înlănțuită în așa mod, ca numerele care coincid după ultimele două cifre să fie într-o sublistă. Se va alege, în calitate de funcție indexată, expresia de tipul: *G(K)=K%100+1*, în calitate de listă de indici *X* – tabloul de 100 de elemente. Pentru lista de indici se poate utiliza metoda stocării indexate. Fie, de exemplu, lista *B* cu elementele: K1=(338, Z), K2=(145, A), K3=(136, H), K4=(214, I), K5=(146, C), K6=(334, Y), K7=(333, P), K8=(127, G), K9=(310, O), K10=(322, X).
30. Pentru două liste date se cere de a le diviza în 8 subliste, adică *X*, în așa mod, ca fiecare sublistă din *B1, B2, ..., B7* să fie elemente, care coincid în prima componentă cu primele două cifre. Lista *X*, la rândul său, va conține indici la lista de indici *Y*, astfel, încât în fiecare sublistă *Y1, Y2, Y3* să se conțină elemente din *X*, la care în prima componentă coincid primele două cifre. Dacă listele *B1, B2, ..., B7* vor fi păstrate înlănțuit, iar listele de indici *X, Y* indexat, astfel de păstrare a listei *B* se va numi stocare indexată consecutiv– înlănțuită. Să se alcătuiască un program de implementare a acestei metode.
31. Sa se implementeze un set de funcții care sa realizeze următoarele operații:
    - adaug (t,id)- introduce identificatorul id în tabela de simboluri t;
    - prezent (t,id)- returnează 1 sau 0 după cum identificatorul id este sau nu prezent în tabel t;
    - sterg (t,id)- elimina id din tabela t;
    - reuniune (t1,t2,t)- t va conține identificatorii prezenți în t1 sau t2;
    - intersectie (t1,t2,t)- t va conține identificatorii prezenți atât în t1 cât și în t2;
    - diferenta (t1,t2,t)- t va conține identificatorii prezenți în t1 și absenți în t2;
    - scriu(t)- tipărește în ordine alfabetica identificatorii din t.

    Folosind funcțiile de mai sus să se realizeze un program care citește două secvențe consecutive de text, care se termina fiecare cu caracterul „.”. După citirea textelor se cere sa se tipărească, în ordine alfabetica, identificatorii prezenți doar în primul text, apoi cei doar în al doilea text. În continuare se vor tipări în ordine alfabetica identificatorii care sunt prezenți atât în primul cât și în al doilea text. În final se afișează identificatorii care apar în cel puțin unul din cele doua texte.
32. Se citește o expresie în notație poloneză, care conține ca operanzi constante reale, iar ca operatori +, -, * și /. Fiecare expresie apare pe o linie separată. După ce s-a citit expresia, ea este evaluata și rezultatul ei este tipărit. Programul citește expresii și le evaluează până la citirea unui 0 singular pe o linie.

Observații: Ca stiva se folosește tipul abstract **stiva** cu valori reale, definit anterior.

- Se va defini funcția getop (char *s), care citește începând cu poziția curenta a liniei de intrare și detectează următorul operand sau operator. *getop* sare peste spatii și caracterele tab. Pentru un operand, funcția returnează constanta *NUMAR* si, în parametrul *s*, adresa de început a șirului format din cifrele numărului. Pentru un operator, *getop* returnează caracterul citit, iar în parametrul *s* adresa de început a șirului format din acel caracter.
- Programul va fi împărțit în trei fișiere: **stiva.h** - conține definițiile constantelor și a tipurilor de date folosite în program, precum și declarațiile funcțiilor ce prelucrează tipul abstract *stiva*. **stiva.c** - conține implementarea tipului abstract *stiva*. **main.c** - conține programul principal precum și funcția *getop*.

33. Sa se realizeze un program C ce tine evidenta personalului unei companii de dimensiuni mici (aproximativ 50 de angajați). Informația referitoare la angajați este păstrată într-un fișier și este folosita pentru inițializarea bazei de date. Fișierul conține linii de forma: nume vârsta adresa numar_matricol funcție

Câmpurile sunt separate printr-un spațiu și sunt șiruri de caractere cu următoarele lungimi: nume - 19, vârsta - 2, adresa - 11, numar_matricol - 5 și funcție - 5. Funcțiile care trebuie implementate sunt:

- *IncarcBD(fis_bd)* - încarcă baza de date din fișierul *fis_bd*.
- *SalvezBD(fis_bd)* - salvează baza de date din memorie în fișierul *fis_bd*.
- *AfisezBD()* - afișează baza de date din memorie.
- *IntroducPersoana(n, v, a, nm, f)* - introduce în baza de date o persoana și datele aferente ei.
- *CautPersoana(n)* - verifica prezenta unei persoane în baza de date.
- *StergPersoana(n)* - șterge din baza de date persoana cu numele *n*.
- *ModificPersoana(n)* - permite modificarea informațiilor legate de persoana cu numele *n*.
- *RetVarsta(n)* - returnează vârsta persoanei cu numele *n*.
- *RetAdresa(n)* - returnează adresa persoanei cu numele *n*.
- *RetNrMatricol(n)* - returnează numărul matricol al persoanei cu numele *n*.
- *RetFunctia(n)* - returnează funcția persoanei cu numele *n*.

34. Elaborați fragmentul programului pentru obținerea listei (fig.1), declarând variabilele dinamice VD și toate celelalte variabile auxiliare cu instrucțiunile necesare pentru obținerea valorilor conform diagramei următoare:

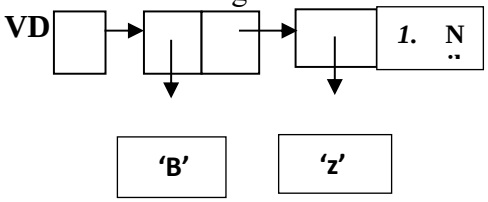

fig.1.

35. Scrieți instrucțiunile ce transformă lista existentă (fig.2) într-un mod nou (fig.3), distrugând variabilele dinamice inutile

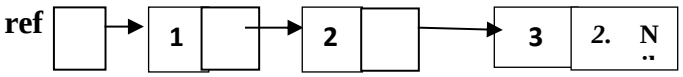

fig.2.

fig.3.