Colegiul Naţional „Gheorghe Lazăr”

București, sector 5

Lucrare pentru obtinerea atestatului profesional la informatica

Soft educațional: liste, stive, cozi

(limbajul C++)

Elev: Profesor coordonator:

Cavaler Alexandru Cristian Camelia Obreja clasa a XII-a A

2022

Cuprins:

1. Motivarea temei..........................................................................................1

2. Noțiuni generale..........................................................................................1

3. Structuri de date .........................................................................................1

3.1 Structura de date de tip listă......................................................................1

3.2 Structura de date de tip stiva.....................................................................3

3.3 Structura de date de tip coada...................................................................4

4. Prezentare program.....................................................................................6

5. Bibliografie ................................................................................................10

6. Anexa.........................................................................................................11

**1. Descrierea temei**

Am ales tema ,,liste, stive, cozi” din dorinţa de intelege cat mai bine acest subiect. Tema are la baza lucrul cu datele sub forma unui vector.

Văd această temă fiind folositoare în dezvoltarea cunoștiințelor generale și totodată fiind un exercițiu în ajutarea memorării datelor ce stau la baza informaticii.

Consider că e important să cunoaștem date despre liste, stive si cozi deoarece ne dezvolta cunostintele despre programare.

**2. Notiuni generale**

O **structură de date** este o colecție de elemente asupra căreia se pot efectua anumite operații.

**Clasificarea structurilor de date:**

a) în funcție de tipul datelor memorate în cadrul structurii:

1. • Structuri de date **omogene** – toate datele componente sunt de același tip (de exemplu tabloul);
2. • Structuri de date **neomogene** – pot conține date de tipuri diferite (de exemplu înregistrarea).
3. b) în funcție de modul de alocare a memoriei interne:
4. • Structuri de date **statice** – ocupă o zonă de memorie de dimensiune fixă, alocată pe întrega durată de execuție a blocului în care este declarată (de exemplu tabloul, fișierul, lista, stiva, coada);
5. • Structuri de date **dinamice** – ocupă o zonă de memorie de dimensiune variabilă, alocată pe parcursul execuției programului, la cererea explicită a programatorului (de exemplu lista, stiva, coada).

**3. Structuri de date**

Structurile de date sunt de 3 tipuri:

1. de tip lista
2. de tip stiva
3. de tip coada

**3.1. Structura de date de tip listă**

**Lista** este o structură de date de tip secvențial, constituită dintr-o succesiune se elemente de același tip. Fiecare lement din listă are un succcesor (cu excepția ultimului element din listă) și un predecesor (cu excepția primului element din listă).

**Obs:** Utilizarea listelor se impune ori de câte ori este necesară organizarea într-o formă secvențială a unui ansamblu de informații.

1. **Operații elementare ce se pot efectua asupra unei liste:**
2. • crearea unei liste vide;
3. • inserarea unui element în listă;
4. • eliminarea unui element din listă;
5. • accesarea unui element din listă;
6. • afişarea elementelor unei liste.
7. **Reprezentarea listelor**: cel mai simplu mod de a implementa o listă constă în memorarea elementelor sale într-un vector.

*Exemplu:* lista cu următoarele elemente: 14, 2, 6, 78.

|  |  |  |  |
| --- | --- | --- | --- |
| 14 | 2 | 6 | 78 |

**O modalitate** deja cunoscută **de implementare a listelor** este prin intermediul [tablourilor statice](https://www.pbinfo.ro/?pagina=articole&subpagina=afisare&id=5583), obținându-se astfel **liste liniare secvențiale**.

Avantajele utilizarii cu liste liniare secventiale:

• accesul la un anumit element se face prin indicele acestuia și este foarte rapid;

• tablourile sunt zone contigue de memorie – elementele sunt alocate în memorie în zone învecinate;

• elementele listei conțin numai informațiile utile.

Dezavantajele utilizarii cu liste liniare secventiale:

• operațiile de [inserare și ștergere](https://www.pbinfo.ro/?pagina=articole-afisare&id=5584) a elementelor presupun parcurgerea tabloului, ceea ce duce la algoritmi lenți;

• dimensiunea tablourilor (numărul de elemente) este precizat la compilare, iar la execuție pot să apară următoarele situații: • spațiul alocat pentru tablou poate fi insuficient;

• spațiul alocat pentru tablou poate fi mult mai mare decât este necesar.

O altă modalitate de implementare a listelor este sub forma **listelor liniare alocate dinamic**. În acest caz, fiecare element al listei este o variabilă dinamică; aceasta va conține, pe lângă **informația utilă** și **informația de legătură**, adică adresa elementului succesor și, eventual, adresa elementului precedent. Sigur, aceste adrese vor fi memorate prin intermediul [pointerilor](https://www.pbinfo.ro/?pagina=articole-afisare&id=7659).

***Exemplu:*** prelucrarea unei liste:

**1.Declararea listei:**

<tip> <identificator>[<nr\_elemente>];

int lista[10];

**2. Crearea unei liste vide**

- se iniţializează numărul de elemente din listă cu 0;

int n=0;

**3. Inserarea unui element în listă**

- se deplasează cu o poziţie la dreapta toate elementele din listă;

- se inserează elementul;

- creşte cu o unitate numărul de elemente ale listei;

for(i=n;i>=poz;i--)

lista[i+1]=lista[i];

lista[poz]=e;

n++;

**4. Eliminarea unui element din listă**

- se deplasează cu o poziţie la stânga toate elementele din listă, începând cu poziţia elementului care se elimină;

- scade cu o unitate numărul de elemente ale listei;

e=lista[poz];

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

lista[i]=lista[i+1];

n--;

**5. Accesarea unui element din listă**

- se memorează un element al listei într-o variabilă;

e=lista[poz];

**6. Afişarea elementelor din listă**

- se parcurge lista element cu element;

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

cout<<lista[i]<<' ';

**3.2 Structuri de date de tip stivă**

**Stiva** este un tip particular de listă, pentru care atât operaţia de inserare a unui element în structură, cât şi operaţia de extragere a unui element se realizează la un singur capăt, denumit vârful stivei.

- singurul element din stivă la care avem acces direct este cel de la vârful stivei;

- trebuie cunoscut în permanenţă vârful stivei;

- stiva este utilizată atunci când programul trebuie să amâne execuţia unor operaţii, pentru a le executa ulterior, în ordinea inversă apariţiei lor;

- stiva funcţionează după principiul **LIFO** (Last In First Out);

**Operaţii elementare care se pot efectua asupra unei stive:**

• crearea unei stive vide;

• înserarea unui element în stivă (operatia se numeste PUSH);

• extragerea unui element din stivă (operatia se numeste POP);

• accesarea elementului de la vârful stivei (operația se numește TOP).

**Reprezentarea stivelor:** cel mai siplu mod de a implementa o stivă constă în memorarea elementelor sale într-un vector.

Diagram

Description automatically generated*Exemplu*: stiva 14, 2, 6, 77

**Utilizarea stivei în programare:**

În programe putem folosi stiva atunci când vrem să amânăm efectuarea unor operații până la

obținerea unor rezultate. De exemplu, conversia unui număr din baza 10 în baza 2 constă în

efectuarea succesivă a unor împărțiri la 2. Cifrele reprezentării în baza 2 sunt resturile împărțirii

în ordine inversă. Ne putem imagina că la fiecare împărțire plasăm restul pe o stivă. În final

golim stiva și afișăm valorile întâlnite.

***Exemplu:*** prelucrarea unei stive:

**1. Declararea stivei:**

<tip> <identificator>[<nr\_elemente>];

int stiva[10];

**2. Crearea unei stive vide**

- se iniţializează numărul de elemente din stivă cu 0;

int vârf=0;

**3. Inserarea unui element în stivă (push)**

- se verifică dacă stiva nu este plină;

- se măreşte vârful stivei;

- se plasează la vârf noul element;

if(varf==10)

cout<<“Stiva este plină”;

else

{ varf++;

stiva[varf]=e; }

**4. Extragerea unui element din stivă (pop)**

- se verifică dacă stiva nu este vidă;

- se reţine elementul din vârful stivei într-o variabilă;

- se micşorează cu o unitate vârful stivei;

if(varf==0)

cout<<“Stiva este vidă”;

else

{ e=stiva[varf];

varf--; }

**5. Accesarea elementului din vârful stivei (top)**

- se memorează elementul din vârful stivei într-o variabilă;

e=stiva[varf];

**3.3 Structura de date de tip coadă**

**Coada** este un tip particular de listă, pentru care operaţia de inserare a unui element se realizează la un capăt, iar operaţia de extragere a unui element se realizează la celălalt capăt.

- singurul element din coadă la care avem acces direct este cel de la început;

- trebuie cunoscut în permanenţă începutul cozii şi sfârşitul cozii;

- coada este utilizată atunci când informaţiile trebuie prelucrate exact în ordinea în care au fost introduse şi ele sunt reţinute în coadă până când pot fi prelucrate;

- coada funcţionează după principiul **FIFO** (First In First Out).

**Operaţii elementare care se pot efectua asupra unei cozi:**

• crearea unei cozi vide;

• înserarea unui element în coadă;

• eliminarea unui element din coadă;

• accesarea unui element din coadă.

**Utilizarea cozii în programare**

Utilizăm coada atunci când trebuie să prelucrăm informații într-o ordine precisă, pe măsură ce

acestea sunt identificate. Uneori, informațiile noi sunt determinate pe baza celor vechi, deja

existente în coadă.

**Reprezentarea cozilor:** cel mai simplu mod de a implementa o coadă constă în memorarea elementelor sale într-un vector.

*Exemplu:* coada14, 2, 6, 14, 77Table

Description automatically generated

**Operatii cu coada:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *1* | *2* | *3* | *4* | *5* |  |
|  |  |  |  |  | Coadă vidă |
| **14** |  |  |  |  | Ad(14) |
| 14 | **2** |  |  |  | Ad(2) |
| 14 | 2 | **6** |  |  | Ad(6) |
| 14 | 2 | 6 | **14** |  | Ad(14) |
| 14 | 2 | 6 | 14 | **77** | Ad(77) (coadă plină) |
| 2 | 6 | 14 | 77 |  | El |
| 6 | 14 | 77 |  |  | El |

**Modalități de implementare a cozii:**

La fel ca stiva, și coada poate fi implementată în limbajul C++ în mai multe moduri:

• implementare statică, prin intermediul tablourilor;

• implementare dinamică, prin intermediul listelor alocate dinamic;

• folosirea containerului queue din STL.

**Implementarea statică a cozii**: utilizează un tablou unidimensional alocat static și două variabile simple prin care identificăm începutul (st) și sfârșitul cozii (dr). Numele variabilelor provine de la **stânga** și **dreapta**, deoarece adăugarea unui element în coadă se face adăugând un element în tablou, iar eliminare se face mărind variabila st – ignorând elementele din față, fără a le șterge efectiv.

***Exemplu:*** prelucrarea unei cozi:

**1.Declararea cozii:**

<tip> <identificator>[<nr\_elemente>];

int coada[10];

**2. Crearea unei cozi vide**

- se iniţializează numărul de elemente din coadă cu 0, pentru aceasta se iniţializează variabilele început şi sfârşit;

int st=1, dr=0;

**3. Inserarea unui element în coadă**

- se verifică dacă coada nu este plină;

- se măreşte sfârşitul cozii cu o unitate;

- se plasează la sfârşit noul element;

if(dr==10)

cout<<“Coada este plină”;

else { dr++;

coada[dr]=e; }

**4. Eliminarea unui element din coadă**

- se verifică dacă coada nu este vidă;

- se reţine elementul de la începutul cozii într-o variabilă;

- se măreşte cu o unitate începutul cozii;

if(st>dr)

cout<<“Coada este vidă”;

else

{ e=coada[st];

st++; }

**5. Accesarea unui element din coadă**

- se memorează elementul de la începutul cozii într-o variabilă;

e=coada[st];

**4. Prezentarea programului**

Operațiile realizate de program sunt distribuite în subprograme, atat de tip void cat si subprograme cu tip.

Biblioteca iostream: permite realizarea operatiilor de citire si afisare pe ecran.

Biblioteca fstream: pentru realizarea operatiilor cu fisiere. Avantajul lucrului cu fisiere este salvarea datelor de intrare si iesire dupa executarea unui program.

Toate datele de intrare le-am citit din fisier.

Am afisat lista, stiva si coada, atat la inceput, cat si dupa fiecare operatie de adaugare/stergere.

Datele le-am afisat pe ecran cu ajutorul comenzii switch pentru realiza un număr nelimitat de operații, fără a fi necesară repornirea programului.

Utilizatorul are posibilitatea de a selecta comanda de afisare dorita.

Pentru citire, afisare, adaugare sau stergere am folosit functia void.

Pentru suma am folosit functia int.

Pentru verificare, daca este vida sau plina, am folosit instructiune din tip if.

Pentru extragerea elementului maxim/minim am folosit instructiune if.

Ceea ce se afiseaza pe ecran dupa ce il rulam:

Prima data se afiseaza lista, stiva si coada, apoi 18 operatii cu acestea

A picture containing text

Description automatically generated

**Pentru lista:**

A screenshot of a computer

Description automatically generated with medium confidence

adauagrea elementui 9 pe pozitia 4

A screenshot of a computer

Description automatically generated with medium confidence

stergerea unui element de pe pozitia 3

A screenshot of a computer

Description automatically generated with medium confidence

verificare daca stiva este plina sau goala

**Pentru stiva:**

A screenshot of a computer

Description automatically generated with medium confidence

adaugarea elementul 8 in stiva

A screenshot of a computer screen

Description automatically generated with medium confidence

stergerea ultimelor 2 elemente din stiva

Text

Description automatically generated

afisarea elementul maxim din stiva:8

**Pentru coada:**

Text

Description automatically generated

verificare daca coada este plina sau goala

A screenshot of a computer

Description automatically generated with medium confidence

adauagrea elementui 6 in coada

A screenshot of a computer

Description automatically generated with medium confidence

stergerea din coada

A screenshot of a computer

Description automatically generated with medium confidence

suma elementelor din coada: 27

A screenshot of a computer

Description automatically generated with medium confidence

elementul maxim din coada: 1

testul:

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

date autor

program final

**6. Bibliografie**

• Teoria predata la orele de clasa, fie din caiet, fie din classroom.

• Manual informatica clasa a XI matematica-informatica de Mariana Milosescu,Editura [Didactica si Pedagogica](https://www.librariadelfin.ro/editura/didactica-si-pedagogica--i11)

• <https://www.pbinfo.ro/articole/13260/liste-liniare-notiuni-introductive>

• <https://www.pbinfo.ro/articole/19577/stiva>

• <http://ase.softmentor.ro/StructuriDeDate/Fisiere/05_CoziStive.pdf>

• <https://www.pbinfo.ro/articole/19579/coada>

• <http://elf.cs.pub.ro/sda-ab/wiki/laboratoare/laborator-04>

**7. Anexa**

Datele de intrare din:

a) fisierul “lista.in”

6

2 3 1 4 8 5

b) fisierul “date.in”

6

6

2

3

1

4

8

5

**Codul programului:**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int lista[21],n,i,stiva[21],vf,coada[10],st=1,dr=0;

void citire\_lista(int &n,int lista[21])

{ ifstream f("lista.in");

f>>n;

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

f>>lista[i];

f.close();}

void afisare\_lista(int n,int lista[42])

{ if (n!=0)

{ cout<<"lista este: ";

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

cout<<lista[i]<<" "; }

else cout<<"lista este vida";

}

void adaugare(int &n,int lista[21])

{ int e,k,i;

do{cout<<"adauga element pe pozitia: ";cin>>k;}while(k<1||k>n+1);

cout<<"elementul: ";cin>>e;

for(i=n;i>=k;i--)

lista[i+1]=lista[i];

lista[k]=e;

n++;

cout<<"noua ";afisare\_lista(n,lista);}

void stergere(int &n,int lista[21])

{ int y;

do{cout<<"sterge element de pe pozitia: ";cin>>y;}while(y<1||y>n+1);

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

lista[i]=lista[i+1];

n--;

cout<<"noua ";afisare\_lista(n,lista); cout<<endl;}

void citire\_stiva(int &vf, int stiva[21])

{ ifstream f("date.in");

f>>vf;

for(int i=1; i<=vf; i++)

{ f>>stiva[i];}

f.close();}

void afisare\_stiva(int vf,int stiva[21])

{cout<<"stiva este: "<<endl;

for(int i=1;i<=vf;i++)

{cout<<"stiva["<<i<<"]=";

cout<<stiva[i];

cout<<endl; }}

void citire\_coada(int &n,int coada[10])

{

ifstream f("date.in");

f>>n;

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

f>>coada[i];

f.close();}

void afisare\_coada(int dr,int coada[10])

{ if(dr!=0)

{cout<<"coada este: ";

{for(int i=st;i<=dr;i++)

cout<<coada[i]<<" ";

cout<<endl;}}

else

cout<<"coada este vida";}

int suma(int n,int coada[10])

{int s=0;

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

s=s+coada[i];

return s;}

void stergere\_coada(int &n,int coada[10])

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

coada[i]=coada[i+1];

n--;afisare\_coada(n,coada); cout<<endl;

}

int main()

{ int n,i,lista[21],k,y,stiva[10],e,p,vf,coada[10],minim=100,maxim=0,st,dr,nr=0;

char r;

citire\_lista(n,lista);

citire\_stiva(vf,stiva);

citire\_coada(n,coada);

afisare\_lista(n,lista);cout<<endl;

afisare\_stiva(vf,stiva);

afisare\_coada(n,coada);

cout<<endl<<"operatii ce pot fi efectuate cu lista:"<<endl;

cout<<"1. afisare lista"<<endl;

cout<<"2. adaugare element in lista"<<endl;

cout<<"3. stergere element in lista"<<endl;

cout<<"4. verificare daca lista este vida"<<endl;

cout<<"5. verificare daca lista este plina"<<endl<<endl;

cout<<"operatii ce pot fi efectuate cu stiva:"<<endl;

cout<<"6. afisare stiva"<<endl;

cout<<"7. adaugare element in stiva"<<endl;

cout<<"8. stergere element din stiva"<<endl;

cout<<"9. extragere element maxim din stiva"<<endl;

cout<<"10. verificare daca stiva este vida"<<endl;

cout<<"11. verificare daca stiva este plina"<<endl<<endl;

cout<<"operatii ce pot fi efectuate cu cozi:"<<endl;

cout<<"12. afisare coada"<<endl;

cout<<"13. adaugare element in coada"<<endl;

cout<<"14. stergere element din coada"<<endl;

cout<<"15. suma elementelor din coada"<<endl;

cout<<"16. elementul minim din coada"<<endl<<endl;

cout<<"17. incepe un test "<<endl<<endl;

cout<<"18. detalii despre autor"<<endl<<endl;

do{cout<<endl<<"alege optiunea: ";cin>>p;

switch(p)

{case 0: return 0;

case 1: {afisare\_lista(n,lista);break;}

case 2: {adaugare(n,lista); cout<<endl;break;}

case 3: {stergere(n,lista); cout<<endl;break;}

case 4: { if (n==0)

cout<<"lista e vida";

else

cout<<"lista nu e vida";

cout<<endl;break;}

case 5: {if (n==20)

cout<<"lista e plina";

else

cout<<"lista nu e plina";

cout<<endl;break;}

case 6: {cout<<" stiva este: "<<endl;

afisare\_stiva(vf,stiva);break;}

case 7: {cout<<"adauga element: ";

if(vf==20)

cout<<"stiva este plină";

else { cin>>e;

vf++;

stiva[vf]=e;}

afisare\_stiva(vf,stiva);break;}

case 8: { cout<<"se sterge un element: ";cout<<endl;

if(vf==0)

cout<<"stiva este vidă";

else{ e=stiva[vf];

vf--;}

afisare\_stiva(vf,stiva);break;}

case 9: {for(int i=1;i<=vf;i++)

{ if(stiva[i]>maxim)

maxim=stiva[i];}

cout<<"elementul maxim este:"<<maxim;

;break;}

case 10: { if (vf==0)

cout<<"stiva e vida";

else cout<<"stiva nu e vida";

cout<<endl;break;}

case 11: {if (vf==20)

cout<<"stiva e plina";

else

cout<<"stiva nu e plina";

cout<<endl;break;}

case 12: {afisare\_coada(n, coada);break;}

case 13: {

if(n==10)

cout<<"coada este plina";

else

{ cout<<"adaugare element: ";

cin>>e;

n++;

coada[n]=e;}

afisare\_coada(n,coada);break;}

case 14: {stergere\_coada(n,coada);

cout<<endl;break;}

case 15: {cout<<"suma= "<<suma(n,coada)<<endl;break;}

case 16: {for(int i=1;i<=n;i++)

{ if(coada[i]<minim)

minim=coada[i];}

cout<<"elementul minim este: "<<minim;

;break;}

case 17:{ cout<<"testul contine 5 intrebari"<<endl;

cout<<"la acest test poti raspunde doar cu 'a'(adevarat) si 'f'(fals)"<<endl<<endl;

cout<<"intrebari: "<<endl;

cout<<"1: ai afisat lista si stiva?"<<endl;///1

do{ cout<<"raspunsul tau: ";cin>>r;}while(r!='a' && r!='f');

if(r=='a')

{nr++;

cout<<"bravo"<<endl;}

else if(r=='f')

{cout<<"asta este"<<endl;}

cout<< "apasa ENTER ca sa continui";

cin.ignore(20, '\n');

cin.get();

cout<<"2: lista nu este vida?"<<endl;///2

do{ cout<<"raspunsul tau: ";cin>>r;}while(r!='a' && r!='f');

if(r=='a')

{nr++;

cout<<"bravo"<<endl;}

else

if(r=='f')

{cout<<"asta este"<<endl;}

cout<< "apasa ENTER ca sa continui";

cin.ignore(20, '\n');

cin.get();

cout<<"3: ai sters minim un element din stiva?"<<endl;///3

do{ cout<<"raspunsul tau: ";cin>>r;}while(r!='a' && r!='f');

if(r=='a')

{nr++;

cout<<"bravo"<<endl;}

else

if(r=='f')

{cout<<"asta este"<<endl;}

cout<< "apasa ENTER ca sa continui";

cin.ignore(20, '\n');

cin.get();

cout<<"4: ai adaugat un element in coada?"<<endl;///4

do{ cout<<"raspunsul tau: ";cin>>r;}while(r!='a' && r!='f');

if(r=='a')

{nr++;

cout<<"bravo"<<endl;}

else

if(r=='f')

{cout<<"asta este"<<endl;}

cout<< "apasa ENTER ca sa continui";

cin.ignore(20,'\n');

cin.get();

cout<<"5: ai calculat suma elementelor din coada?"<<endl;///5

do{ cout<<"raspunsul tau: ";cin>>r;}while(r!='a' && r!='f');

if(r=='a')

{nr++;

cout<<"bravo"<<endl;}

else

if(r=='f')

{cout<<"asta este"<<endl;}

cout<<"ai obtinut nota "<<nr<<" din 5";break;}

case 18:{ cout<<" elev: Cavaler Alexandru"<<endl;

cout<<" clasa XIIA";break;}

default:cout<<"valoare gresita!";

}}while(p!=0);

return 0;}