**Liste liniare dublu inlantuite**

O componenta a unei liste dublu inlantuite se declara ca o data structurata de tip inregistrare, formata din trei campuri: informatia propriu-zisa (care poate fi de orice tip: numeric, caracter, pointer, tablou, inregistrare) si informatiile de legatura (adresa la care e memorata urmatoarea componenta si adresa la care e memorata precedenta componenta). Ultima componenta va avea informatia de legatura corespunzatoare urmatoarei adrese NULL (sau 0), cu semnificatia ca dupa ea nu mai urmeaza nimic (retine adresa „nici o adresa” a urmatoarei componente). La fel si in cazul primei componente pentru campul adresa precedenta.





Conform celor enuntate anterior memorarea si prelucrarea acestor structuri de date este similara cu cea a listelor liniare simplu inlantuite, ba chiar unele prelucrari (cum ar fi stergerea si inserarea inainte se simplifica avand intr-un mod facil acces la componenta anterioara)

**Operatii specifice**

* Creare
* Parcurgere (afisare)
* Cautare element
* Inserare element
* Stergere element

Declaratii globale

**struct nod**

**{ int inf; // informatia utila**

**nod \*urm, \*pred; // adresa urmatorului element si al al elementului predecesor**

**} \*p, \*u;**

// p= adresa primului nod al listei

// u= adresa ultimului nod al listei

1. **Crearea unei liste dublu inlantuite cu n elemente citite de la tastatura**

void creare (nod \*&p, nod \*&u)

{ int n,i;

nod \*c;

cin>>n;

p=u=NULL;

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

{ c=new nod; // alocam spatiu la adresa c pentru noul nod

cin>>c->inf; // citim informatia utila de la adresa c

**c->pred**=c->urm=NULL; // ii completez cu NULL adresele de legatura

if (p==NULL) p=u=c; // daca lista e vida primul si ultimul nod devin c

else { u->urm=c; // legam c dupa u – urmatorul de dupa u devine c

**c->pred=u; // predecesorul lui c devine u**

u=c; // actualizam adresa ultimului nod cu c

}

}

}

1. **Afisarea (parcugerea) elementelor din lista**
2. **De la primul la ultimul (de la stanga la dreapta)**

void afis (nod \*p)

{ nod \*c;

c=p;

while (c!=NULL) // while(c)

{ cout<<c->inf<<” ”; // afisam informatia utila de la adresa c

c=c->urm; // se avanseaza la urmatoarea adresa

}

cout<<endl;

}

**Sau**

for (c=p; c!=NULL; c=c->urm) cout<<c->inf<<” ”;

1. **De la ultimul catre primul (de la dreapta la stanga)**

**for(c=u; c; c=c->pred) cout<<c->inf<<” ”**;

1. **Cautarea unui element x in lista**

int caut (nod \*p, int x)

{nod \*c;

c=p;

while (c && c->inf!=x) c=c->urm;

if (c==NULL) return 0;

else return 1;

}

1. **Adaugarea unui nou element x la sfarsitul listei (dupa ultimul)**

void adaug (nod \*&p, nod \*&u, int x)

{ nod \*c;

c=new nod; // aloc spatiu la adresa c pentru noul nod

c->inf=x; // completez informatia utila de la adresa c

**c->pred**=c->urm=NULL; // ii completez cu NULL adresele de legatura

if (p==NULL) p=u=c; // daca lista e vida primul si ultimul nod devin c

else { u->urm=c; // leg c dupa u – urmatorul de dupa u devine c

**c->pred=u; // predesorul lui c devine u**

u=c; // actualizez adresa ultimului nod cu c

}

}

1. **Adaugarea unui nou element x la inceputul listei (inainte de primul)**

void adaug (nod \*&p, nod \*&u, int x)

{ nod \*c;

c=new nod;

c->inf=x;

c->urm=**c->pred=NULL**;

if (p==NULL) p=u=c;

else { c->urm=p; // c -> p

**p->pred=c;**

p=c;

}

}

1. **Inserarea unui nou element dupa elementul x (x diferit de ultimul element)**

**Exemplu : 10, 20, 30, 40, 50 si x=20 si inseram valorea citita 100 ->**

**10, 20, 100, 30, 40, 50**

void inserare( nod \*p, int x)

{ nod \*c, \*d;

c=p;

while (c && c->inf!=x) c=c->urm; // c= adresa lui x sau NULL

if (!c) cout<< x<<” nu e in lista”<<endl;

else { // **c** -> **d** ->  **c->urm**

d=new nod; // alocam spatiu pentru el. care se insereaza

cin>> d->inf; // inseram adresa d intre c inte c->urm

**d->pred=c;**

d->urm=c->urm;

**c->urm->pred=d;**

c->urm=d;

}

}

1. **Inserarea unui nou element inainte de elementul x (x diferit de primul element)**

**Exemplu : 10, 20, 30, 40, 50 si x=30 si inseram valorea citita 100 ->**

**10, 20, 100, 30, 40, 50**

void inserare( nod \*p, int x)

{ nod \*c, \*d;

c=p;

while (c && c->inf!=x) c=c->urm; // c= adresa lui x sau NULL

if (!c) cout<< x<<” nu e in lista”<<endl;

else { // **c->pred** -> **d** ->  **c**

d=new nod;

cin>>d->inf;

d->pred=c->pred;

d->urm=c;

c->pred->urm=d;

c->pred=d;

}

}

1. **Stergerea elementului x din lista ( x diferit de primul si ultimul element)**

void sterg ( nod \*p, int x);

{ nod \*c, \*aux;

c=p;

while (c && c->inf!=x) c=c->urm; // x va fi la adresa c

aux=c;

// **c->pred c c->urm**

c->pred->urm=c->urm;

c->urm->pred=c->pred;

delete aux;

}

1. **Stergerea primului element din lista (nevida)**

void sterg (nod \*&p)

{ nod \*aux;

aux=p;

p=p->urm;

**p->pred=NULL;**

delete aux;

}

1. **Stergerea ultimului element din lista (lista contine c. p. 2 elemente)**

void sterg (nod \*&u)

{ nod \*aux;

aux=u;

**u=u->pred;**

u->urm=NULL;

delete aux;

}

1. **Eliberarea spatiului de memorie ocupat de lista**

void sterg\_lista(nod \*&p)

{nod \*aux;

while (p!=NULL) // sterg primul element

{ aux=p;

p=p->urm;

**p->pred=NULL;**

delete aux;

}

}

1. **Scindarea unei liste in doua liste prin valoarea x ( x va face parte din prima lista)**

**p c u**

**x**

void scindare (nod \*p, nod \*u, int x, nod \*&p1, nod \*&u1, nod \*&p2, nod \*&u2)

{ nod \*c;

c=p;

while (c->inf!=x) c=c->urm;

p1=p;

u1=c; u1->urm=NULL

p2=c->urm; p2->pred=NULL;

u2=u;

}

1. **Concatenarea a doua liste p1, u1 si p2, u2 in p3, u3**

**p1 u1 p2 u2**

void concatenare (nod \*p1, nod \*u1, nod \*p2, nod \*u2, nod \*&p3, nod \*&u3)

{ nod \*c;

p3=p1; u3=u2;

u1->urm=p2;

p2->pred=u1;

}

**Tema:**

1. **Se dau doua liste dublu inlantuite. Sa se verifice daca a doua lista este inversa primei liste.**

**Exemplu:**

**prima lista: 1 2 3 4 5**

**a doua lista: 5 4 3 2 1 -> afiseaza DA**

1. **Stergerea tuturor elementelor prime dintr-o lista dublu inlantuita cu n numere intregi.**
2. **Sa se insereze dupa fiecare numar palindrom dintr-o lista liniara dublu inlantuita opusul sau.**
3. **Fisierul LISTA.IN contine pe prima linie cel mult 1000000 numere intregi separate prin cate un spatiu. Sa se creeze o lista dublu inlantuita cu numerele distincte din fisier si cu frecventele lor. Lista obtinuta va fi afisata in fisierul LISTA.OUT, fiecare numar cu frecventa lui pe cate o linie a fisierului.**

**Exemplu: daca fisierul LISTA.IN contine numerele 7 7 4 5 7 2 3 1 2 5 7 1 5,**

**fisierul LISTA.OUT va contine**

**1 2**

**2 2**

**3 1**

**4 1**

**5 3**

**7 4**

1. **Fisierul IN.TXT contine pe prima linie cel mult 1000000 numere intregi separate prin cate un spatiu. Sa se creeze o lista dublu inlantuita cu numerele din fisier si apoi sa se stearga din ea toate aparitiile valorii maxime din lista. Lista obtinuta va fi afisata in fisierul OUT.TXT.**

**Exemplu: 7 7 4 7 7 7 3 1 2 5 7 7**

* **4 3 1 2 5**