**Lista liniara simplu inlantuita –** o lista liniara in care fiecare element contine pe langa informatia utila si adresa urmatorului element din lista.

1

2

n

NULL

**p u**

**Operatii specifice**

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

**Declaratii globale**

**struct nod**

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

**nod \*urm; // adresa urmatorului element**

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

// p= adresa primului nod al listei

// u= adresa ultimului nod al listei

1. **Crearea unei liste simplu 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; // aloca spatiu la adresa c pentru noul nod

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

c->urm=NULL; // ii completez cu NULL adresa de legatura

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

NULL

NULLL

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

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

}

}

}

1. **Afisarea (parcugerea) elementelor din lista**

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. **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. **Inserarea unui nou element dupa elementul x**

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

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

x

void inserare( nod \*p, int x)

{ nod \*c, \*d; c c->urm

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; d

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

cin>> d->inf;

d->urm=c->urm;

c->urm=d;

}

}

x

1. **Stergerea elementului x din lista**

void sterg ( nod \*&p, int x); c c->urm c->urm->urm

{ nod \*c, \*aux;

if (p->inf==x) // sterg primul element

{aux=p;

p=p->urm;

delete aux;

}

else { c=p;

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

aux=c->urm;

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

delete aux;

}

}

**Tema**: 1168, 1169, 1172, 1173, 1175, 1171

1. **Stergerea ultimului element din lista – cu rezultat ultima adresa**

void sterg\_ultim (nod \*p, nod \*&u)

{ nod \*c, \*aux;

// avansez cu c pana la adresa penultimului el. din lista

c=p;

while (c->urm->urm!=NULL) c=c->urm;

aux=c->urm; //adresa ultimului el. – sau aux=u

u=c;

u->urm=NULL;

delete aux;

}

1. **Adaugarea unui nou element x la sfarsitul listei, cu adresa de inceput p si adresa de sfarsit u**

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->urm=NULL; // ii completez cu NULL adresa 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

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

}

}

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

void sterg\_lista(nod \*&p)

{nod \*aux;

while (p!=NULL)

{ aux=p;

p=p->urm;

delete aux;

}

}

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

**p c c->urm u**

**x**

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

{ nod \*c;

c=p;

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

p1=p; p2=c->urm;

**c->urm=NULL;**

}

1. **Concatenarea a doua liste p1 si p2 in p3**

**p1 c p2**

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

{ nod \*c;

c=p1;

while (c->urm!=NULL) c=c->urm; // c=adresa ultimului el. din prima lista

c->urm=p2;

p3=p1;

}