Pointeri. Alocarea dinamică a memoriei

```
#include <iostream>
using namespace std;
struct elev
{ char nume[20];
  char prenume[20];
  float nota1;
  float nota2;
  int varsta;
};

int main()
{
    elev *a,b,*c;
   a=new elev;
   cin>>(*a).nume;
   cin>>(*a).varsta;
   cin>>a->prenume;

   cout<<a->varsta-10;

   cin>>b.nume;
   cin>>b.varsta;
   cin>>b.prenume;
   cout<<b.varsta-10;

   c=&b;
   cout<<c->varsta-10;
delete a;
    return 0;
}
```

Definiția listei:

O listă este o colecție de noduri aflate într-o relație de ordine. Astfel există primul nod al listei, al doilea nod al listei, ..., ultimul nod al listei.

Operațiile permise sunt:

- accesul la nodurile listei pentru actualizare;
- adăugarea unui nod;
- ștergerea unui nod;
- schimbarea poziției unui nod în listă;

Fiecare nod, cu excepția ultimului, reține adresa nodului următor.

Dezavantaje:

1. Accesul la un nod al listei se face prin parcurgerea nodurilor care îl preced.

2. Informațiile de adresă sunt prezente în cadrul fiecărui nod.

Avantaje:

1. Operațiile de adăugare / ștergere sunt rapide.

Clasificarea listelor:

★ liste liniare simplu/dublu înlănțuite
★ liste circulare simplu/dublu înlănțuite

Liste liniare simplu înlănțuite

Crearea listelor
Afișarea listelor

Exemplul 1:

```
#include <iostream>
using namespace std;

struct nod
{ int info;
  nod* urm;
};

void adaugInc(nod*& v, int nr)
{
    nod* c=new nod;
    c->info=nr;
   c->urm=NULL;
    if(v==NULL) v=c;
    else
    {
     c->urm=v;
     v=c;
    }
}

void tipar(nod* v)
{ nod* c=v;
  while (c)
  { cout<<c->info<<'  ';
    c=c->urm;
  }
}
void initializare(nod *&p)
{
    p=NULL;
}
int main()
{ int x;
  nod *pr;
  initializare(pr);
    cin>>x;
  while (x)
  {   adaugInc(pr,x);
       cin>>x;
   }
   tipar(pr);
```

```
}
```

Exemplul 2:

```
#include <iostream>
using namespace std;

struct nod
{ int info;
  nod* urm;
};

void adaugSf(nod*& p, int nr)
{
    nod* c=new nod;
    c->info=nr
    c->urm=NULL;
    if(p==NULL) p=c;
    else
    {
      nod *u=p;
      while(u->urm!=NULL) u=u->urm;
      u->urm=c;
    }
}

nod* adaugSfRec()
{ nod* p;
  int x;
  cout<<"numar "; cin>>x;
  if (x)
  {
      p=new(nod);
      p->urm=adaugSfRec();
      p->info=x;
      return p;
  }
  else return 0;
}
void tipar(nod* v)
{
  nod* p=v;
  while (p)
  {
     cout<<p->info<<' ';
      p=p->urm;
```

```
  }
}
void initializare(nod *&p)
{
    p=NULL;
}

int main()
{ int x;
  nod *pr1;
  nod *pr2;
  initializare(pr1);
  initializare(pr2);
  pr1=adaugSfRec();
      cin>>x;
 while (x)
 {
    adaugSf(pr2,x);
    cin>>x;
 }
tipar(pr1);
tipar(pr2);
}
```