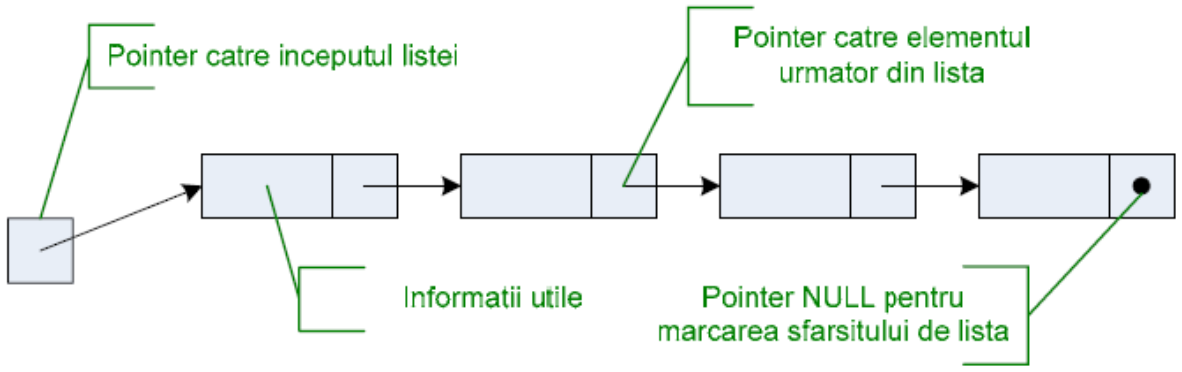
Lista simpla

Listele simplu inlantuite sunt structuri de date dinamice omogene. Spre deosebire de masive, listele nu sunt alocate ca blocuri omogene de memorie, ci ca elemente separate de memorie. ***Fiecare nod al listei contine, in afara de informatia utila, adresa urmatorului element. Aceasta organizare permite numai acces secvential la elementele listei.***

Pentru accesarea listei trebuie cunoscuta adresa primului element (numita capul listei); elementele urmatoare sunt accesate parcurgand lista.

Lista simplu inlantuita poate fi reprezentata grafic astfel:



Lista simpla este o structura dinamica. Se caracterizeaza prin:

- o variabila pointer, care contine adresa unei zone de memorie numita primul element al listei;
- zona de memorie se compune din doua subzone: o zona cu informatii utile si o variabila pointer ce contine adresa elementului urmator;
- celelalte elemente ale listei sunt legate intre ele;
- ultimul element al listei care are variabila pointer initializata cu NULL pentru a marca inexistenta unui element urmator.

Modelul graf presupune:

- o multime de noduri;
- o multime de arce;
- fiecare nod este legat de altul PRINTR-UN SINGUR ARC;
- un nod care nu are un arc incident spre el;
- un nod care nu are nici un arc incident spre exterior.

Modelul text sursa presupune o secventa de definire recursiva de articole.

Modelul analitic presupune existenta elementelor *E1*, *E2*, *E3*,....,*Ex*. Fiecare element are un camp de informatie utila *IU* si un pointer de legatura *PL*. Exista o variabila pointer cu care se refera primul element P1.

$cont(P_1)=adr(E_1)$
$cont(E_i.PL)=adr(E_{i+1})$
$i=1,2,3,..,x-1$
$cont(Ex.PL)=NULL$
$cont(E_i.IU) = sir_i,\ i=1,2,..,x$

Daca elementele listei simple se definesc prin:

```
struct lista_simpla
{
int info;
struct lista_simpla *next;
};
struct lista_simpla a,b,c, *p;
```

si daca se initializeaza corespunzator aceste variabile, atunci referirea elementelor direct folosind pointerul *i* presupune folosirea repetata a operatorului de referire cu variabile pointeri.

1. Traversarea unei liste simplu inlantuite

Daca nodul de inceput al listei este indicat de variabila *inceput*, o variabila auxiliara *q*, care parcurge toate nodurile listei pana cand valoarea ei devine *NULL*, permite accesul la fiecare nod si efectuarea operatiei specifice traversarii:

```
for(q=inceput;q!=NULL;q=q->urmator)
  //prelucrarea nodului indicat de q
```

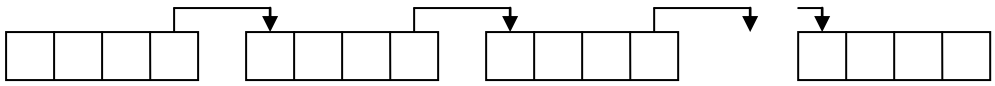
Daca lista are doua noduri fictive, unul de inceput si unul de sfarsit, secventa de traversare devine:

```
for(q=inceput->urmator;q!=sfarsit;q=q->urmator)
  //prelucrarea nodului indicat de q
```

Exemplul 1:

Se considera un magazin in care se afla un numar de *n* produse, identificate prin codurile *c1, c2, ..., cn*. Fiecare produs *ci* se gaseste in cantitatea *qi* si are pretul *pi*. Se cere sa se determine valoarea *vi* a fiecarui produs *i*, precum si valoarea totala a produselor din magazin, tinand cont de faptul ca se utilizeaza o lista simplu inlantuita pentru stocarea valorilor *ci, qi, pi, i=1..n.*

Memorarea valorilor intr-o lista simplu inlantuita:



Exemplul 2:

Sa se realizeze un program C++ pentru crearea unei liste simplu înlănţuite şi:

- ştergerea elementului de pe poziţia “k”;
- ştergerea unui nod cu informaţie dată;
- ştergerea elementelor pare şi impare din listă;
- ştergerea elementelor de pe poziţii pare şi impare;
- stergerea nodului situat înaintea unui nod cu informaţie dată;
- ştergerea întregii liste.

Exemplul 3:

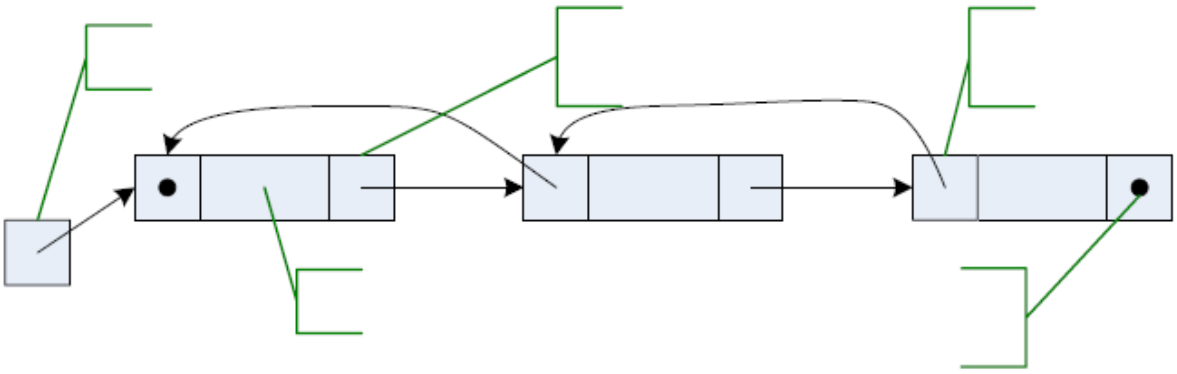
Sa se realizeze un program C++ pentru crearea si afisarea unei liste circulare simplu înlănţuite.

Lista dubla

Spre deosebire de listele simplu înlănţuite care permit parcurgerea de la primul element spre ultimul alocat dinamic, ***listele dublu înlănţuite realizează şi drumul invers, permiţând şi parcurgerea de la ultimul element către primul element***.

Listele dublu inlantuite sunt structuri de date dinamice omogene. ***Ele au aceleasi caracteristici de baza ca si listele simplu inlantuite. Diferenta fata de acestea consta in faptul ca, pentru fiecare nod, se retine si adresa elementului anterior, ceea ce permite traversarea listei in ambele directii.***

Lista dublu inlantuita poate fi reprezentata grafic astfel:



Listele duble sunt structuri de date dinamice care permit traversarea in ambele sensuri.
Fiecare element contine trei componente:
- un pointer care memoreaza adresa elementului precedent
- o structura de tip articol sau un camp in care se memoreaza informatia utila prelucrarii;
- un pointer care memoreaza adresa elementului urmator din structura.

Lista dubla are doi pointeri pentru referirea elementelor ce o compun:
- un pointer pentru a referi elementele cand procesul de traversare se efectueaza de la stanga spre dreapta; acest pointer refera la inceput primul element al listei duble, iar dupa terminarea traversarii refera ultimul element;
- un pointer care permite traversarea de la dreapta spre stanga; acest pointer refera ultimul element din lista dubla; dupa traversare, va referi primul element al listei.