

Lista înlănțuită

- Structură de date *dinamică* - una dintre cele mai simple și des folosite structuri fundamentale de date pentru reprezentarea TAD-urilor (*Colecție*, *Mulțime*, *Dicționar*, *Listă*, *Stivă*, *Coadă*, etc).
 - Listele înlănțuite sunt folosite pentru reprezentarea *înlănțuită* a containerelor de date.
 - *Lista înlănțuită* - colecție de elemente stocate în locații numite *noduri*, a căror **ordine** este determinată de o *legătură* (referință) conținută în fiecare nod.
 - ordine între *pozițiile* elementelor în cadrul listei
 - Fiecare nod al listei conține:
 - elementul propriu-zis (informația utilă) din nodul listei (poate fi văzut ca o cheie a nodului); și
 - legături (referințe):
 - * spre următorul element din listă; ȘI/SAU
 - * spre precedentul element din listă.
 - Caracteristica reprezentării înlănțuite - **accesul secvențial**: un element va putea fi accesat pornind de la primul element al listei (numit și capul listei) și urmând legăturile până când elementul este găsit (operația are complexitate-timp *liniară*).
 - Operații:
 - inserare/ștergere elemente pe orice *poziție* în listă, se poate determina succesorul (predecesorul) unui element aflat pe o anumită *poziție*, se poate accesa un element pe baza *poziției* sale în listă.
 - depind de specificul containerului care se reprezintă folosind lista înlănțuită.
 - Există următoarele tipuri de reprezentări înlănțuite:
1. **Reprezentare simplu înlănțuită. (Singly linked list)** – În fiecare nod este memorată legătura spre următorul element din listă. Lista va fi identificată printr-o referință la primul său element (se poate memora referință către ultimul element pentru a eficientiza anumite operații).
 - Dacă legăturile între noduri ar fi memorate sub formă de adrese în memorie (pointeri), atunci ultimul element va conține în câmpul de legătură pointerul nul (NIL). În acest caz, dacă *prim* este NIL, atunci lista este vidă.
 2. **Reprezentare dublu înlănțuită. (Double linked list)** – În fiecare nod sunt memorate legături atât spre următorul element din listă, cât și spre precedentul element. Lista va fi identificată prin câte o referință la primul și la ultimul său element.

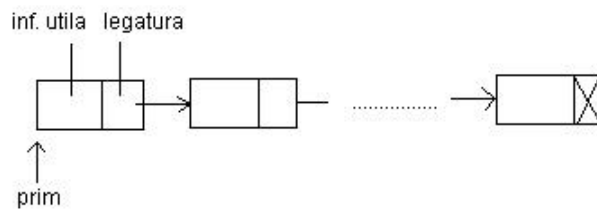


Figura 1: Reprezentarea simplu înlănțuită a listelor.

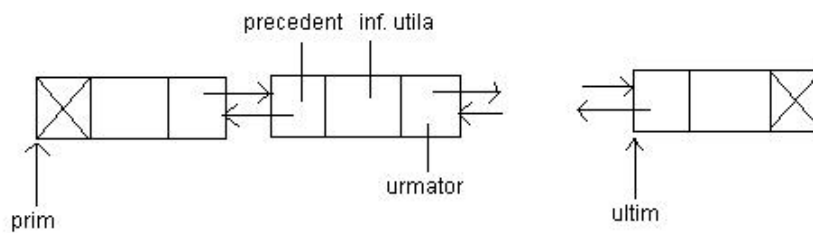


Figura 2: Reprezentarea dublu înlănțuită a listelor.

- Dacă legăturile între noduri ar fi memorate sub formă de adrese (pointeri), atunci ultimul element va conține în câmpul de legătură **urmator** pointerul nul (NIL), iar primul element va conține în câmpul de legătură **precedent** pointerul nul. În acest caz, dacă **prim** este NIL și **ultim** este NIL, atunci lista este vidă.
3. **Reprezentare înlănțuită circulară (Circular linked list)**– *reprezentare înlănțuită (simplu sau dublu)* în care ultimul element are o legătură spre primul element al listei. Lista va păstra o referință către ultimul element al listei - **ultim**; referința către primul nod se obține din legătura spre următorul element al nodului ultim.

Avantaje ale reprezentării înlănțuite

- noi elemente pot fi adăugate sau șterse oriunde în listă, gestionând legăturile între elemente, fără costuri mari (complexități-timp *constante*);
- în cazul în care se folosește alocarea dinamică nu există limitare a capacității listei (număr de elemente).

Dezavantaje ale reprezentării înlănțuite

- spațiu suplimentar de memorie pentru memorarea legăturilor.
- accesul la elementul de pe poziția k este dificil (complexitate timp *liniară*).

Modalități de reprezentare a înlanțuirilor

1. folosind alocare dinamică a memoriei (*poziția* în cadrul listei va fi adresa de memorare a unui nod al listei - *pointer*).
2. folosind alocare statică a memoriei (tablou) (*poziția* în cadrul listei va fi indicele din tablou unde se memorează un nod al listei - *întreg*).

Alte tipuri de liste înlanțuite

- **Liste dublu înlanțuite de tip XOR** (*XOR double linked list*) - pentru a reduce spațiul de memorare pentru a stoca legăturile în nodurile unei liste dublu înlanțuite
- **Skip lists**