Lista înlănţuită

- Structură de date *dinamică* - una dintre cele mai simple şi des folosite structuri fundamentale de date pentru reprezentarea TAD-urilor (*Colecţie*, *Mulţime*, *Dicţionar*, *Listă*, *Stivă*, *Coadă*, etc).
- Structură de date liniară.
- Listele înlănţuite sunt folosite pentru reprezentarea *înlănţuită* a containerelor de date.
- *Lista înlănţuită* - colecţie de elemente stocate în locaţii numite *noduri*, a căror **ordine** este determinată de o *legătură* (referinţă) conţinută în fiecare nod.
  - ordine între *poziţiile* elementelor în cadrul listei
- Fiecare nod al listei conţine:
  - elementul propriu-zis (informaţia utilă) din nodul listei (poate fi văzut ca o cheie a nodului); şi
  - legături (referinţe):
    - spre **următorul** element din listă; ŞI/SAU
    - spre **precedentul** elementul din listă.
- Nodurile unei liste înlănţuite nu sunt memorate în locaţii succesive în memorie (aşa cum se întâmplă la reprezentarea secvenţială)
  - de aceea e necesar să memorăm nodul **următor/precedent**
- Caracteristica reprezentării înlănţuite - **accesul secvenţial**: un element va putea fi accesat pornind de la primul element al listei (numit şi capul listei) şi urmând legăturile până când elementul este găsit (operaţia are complexitate-timp *liniară*).
- Operaţii (depind de specificul containerului care se reprezintă folosind lista înlănţuită):
  - inserare/ştergere elemente pe orice *poziţie* în listă
  - căutarea unei valori în listă
  - determinarea succesorului (predecesoruluil) unui element aflat pe o anumit *poziţie*
  - accesarea unui element pe baza *poziţiei* sale în listă.

Există următoarele tipuri de reprezentări înlănţuite:

1. **Reprezentare simplu înlănţuită.** (**Singly linked list**)

   - În fiecare nod este memorată legătura spre următorul element din listă.
   - Lista va fi identificată printr-o referinţă la primul său element
   - se poate memora şi referinţă către ultimul element pentru a eficientiza anumite operaţii.

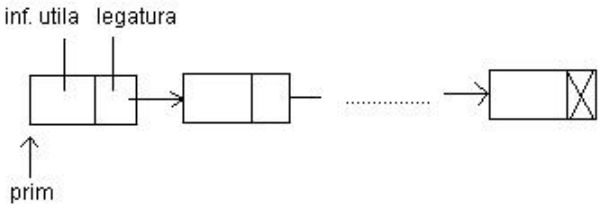

Figura 1: Reprezentarea simplu înlănţuită a listelor.

   - Dacă legăturile între noduri ar fi memorate sub formă de adrese în memorie (pointeri), atunci ultimul element va conţine în câmpul de legătură pointerul nul (NIL).
     - În acest caz, dacă prim este NIL, atunci lista este vidă.

2. **Reprezentare dublu înlănţuită.** (**Double linked list**)

   - În fiecare nod sunt memorate legături atât spre următorul element din listă, cât şi spre precedentul element.
   - Lista va fi identificată prin câte o referinţă la primul şi la ultimul său element.

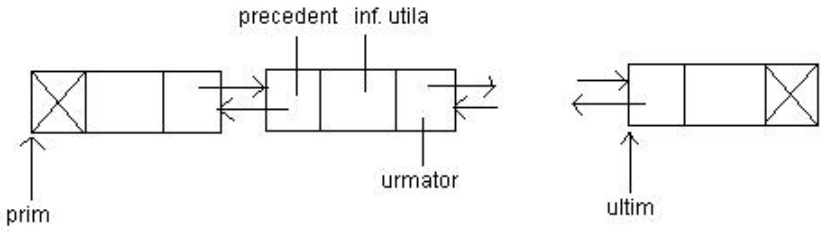

Figura 2: Reprezentarea dublu înlănţuită a listelor.

   - Dacă legăturile între noduri ar fi memorate sub formă de adrese (pointeri), atunci ultimul element va conţine în câmpul de legătură urmator pointerul nul (NIL), iar primul element va conţine în câmpul de legătură precedent pointerul nul.
     - În acest caz, dacă prim este NIL şi ultim este NIL, atunci lista este vidă.

3. **Reprezentare înlănţuită circulară** (**Circular linked list**)– *reprezentare înlănţuită (simplu sau dublu)* în care

   - ultimul element are o legătură spre primul element al listei.
   - Lista va păstra o referinţă către ultimul element al listei - ultim
   - referinţa către primul nod se obţine din legătura spre următorul element al nodului ultim.

Avantaje ale reprezentării înlănţuite

- noi elemente pot fi adăugate sau şterse oriunde în listă, gestionând legăturile între elemente, fără costuri mari (complexităţi-timp *constante*);
  - în Figura 3(a) este ilustrată o adăugare între două noduri într-o *listă simplu înlănţuită* - $\theta(1)$
    - se creează nodul conţinând informaţia utilă dorită (C)
    - se adaugă legăturile în ordinea 1, 2 (marcate cu **roşu**)
  - în Figura 3(b) este ilustrată ştergerea unui nod dintr-o *listă simplu înlănţuită* - $\theta(1)$
    - ştergerea nodului marcat cu albstru presupunere adădŭgarea legăturii evidenţiate pe figură

(a) Adăugare element între două noduri. (b) Ştergere nod.

Figura 3: Exemplu **adăugare** (a) şi **ştergere** (b).

- în cazul în care se foloseşte alocarea dinamică, nu există limitare a capacităţii listei (număr de elemente).

Dezavantaje ale reprezentării înlănţuite

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

Modalităţi de reprezentare a înlănţuirilor

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

Alte tipuri de liste înlănţuite

A se consulta documentul de pe pagina cursului, http://www.cs.ubbcluj.ro/~gabis/sda/Cursuri/Cur
- Liste inlantuite - alocare dinamica/3_XOR Lists and Skip Lists.pdf

- **Liste dublu înlănţuite de tip XOR** (*XOR double linked list*)
  - pentru a reduce spaţiul de memorare pentru a stoca legăturile în nodurile unei liste dublu înlănţuite
- **Skip lists**
  - structură eficientă de date pentru memorarea unui dicţionar ordonat