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).
- 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.
- 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.
- 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:
  - inserare/ştergere elemente pe orice *poziţie* în listă, se poate determina succesorul (predecesorul) unui element aflat pe o anumit *poziţie*, se poate accesa un element pe baza *poziţiei* sale în listă.
  - depind de specificul containerului care se reprezintă folosind lista înlănţuită.
- 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.

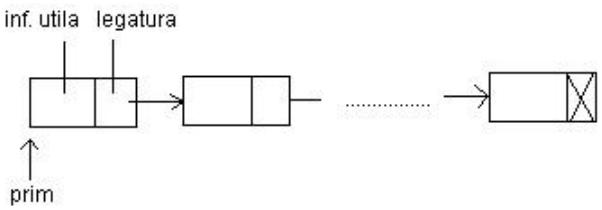

Figura 1: Reprezentarea simplu înlănţuită a listelor.

– Dacă legăturile între noduri ar fi memorate sub formă de adrese (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.

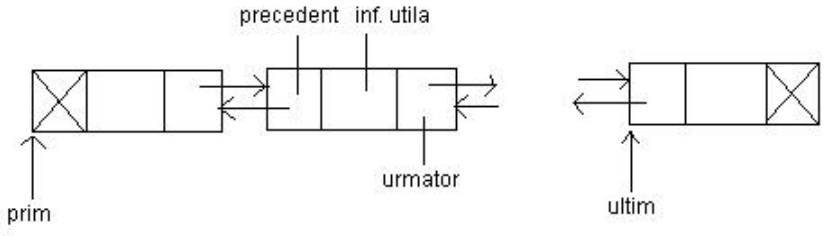

Figura 2: Reprezentarea dublu înlănţuită a listelor.

– 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 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*).

Lista Inlănţuită - reprezentarea înlănţuirilor pe tablou

Să considerăm lista l=(a, b, c, d, e, f) reprezentată simplu înlănţuit, cu înlănţuirile reprezentate folosind un tablou având capacitatea maximă de 10 locaţii.

Reprezentarea este dată mai jos.

Indice	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
e	c	-	e	a	-	b	-	-	d	f
urm	9	5	10	6	8	1	2	0	3	0

Tabela 1: $prim$=4, $primLiber$=7

Observaţii:

1. Capacitatea vectorilor e şi urm poate fi mărită dinamic, dacă este necesar - numărul de elemente din listă depăşeşte numărul de locaţii alocat iniţial (vezi vectorul dinamic).
2. În cazul redimensionării, considerând lista reprezentată simplu înlănţuit, operaţia $adaugaInceput$ are complexitatea timp **amortizată** $\theta(1)$.