

Structuri de date și algoritmi

Curs, IS – An II

```
100101001010
01010110001010010100
01010110001010010100101001
10101100010100101001010010 100
0010101100010100101001010010100 10 10
010101100010100101001010010100111
1000101001010010100101001010
01001010010100101001010
101100
011000
01100
011000
101100
00101
110001010010011
01010110001010010100101001
```

Tablouri și liste

“Nothing is more practical than a good theory.”

(Ludwig Boltzmann)

Tablouri

Un tablou este o structura de date abstracta **omogenă**, **statică**.

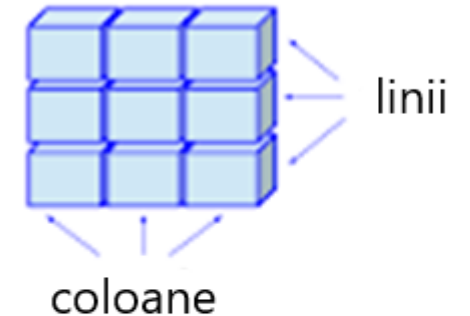
Fiecare element se identifică pe baza unui **index** (unor indecsi) iar timpul de acces este același pentru fiecare element.

- Indecsi aparțin unei mulțimi peste care este definită o relație de ordine totală;
- **Dimensiune fixă** specificată la alocare;
- Elementele ocupă **locații succesive în memorie**;
- Numele tabloului se asociază cu locația de memorie a primului element;
- Adăugarea unui element – operație laborioasă
 1. Crearea unui nou vector de dimensiune mai mare
 2. Copierea elementelor în noul câmp
- Modificarea pozițiilor elementelor (de ex. într-un vector sortat) operație foarte laborioasă.

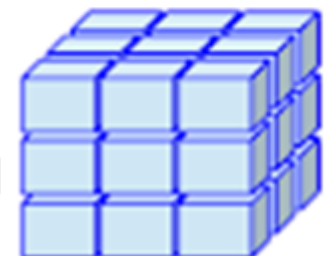
Vector



Matrice



Tabloul multidimensional



Tablouri k dimensionale interpretate ca vectori de tablouri $k-1$ dimensionale

Imagine RGB: Vector de 3 matrice (una pentru fiecare culoare)



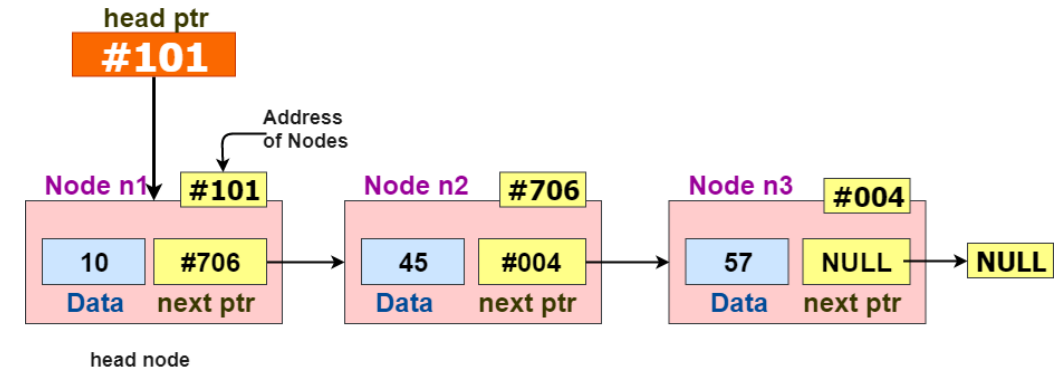
0033 R: 153 G: 000 B: 051	FF3366 R: 255 G: 051 B: 102	CC0033 R: 204 G: 000 B: 051	FF0033 R: 255 G: 000 B: 051	FF9999 R: 255 G: 153 B: 153	CC3366 R: 204 G: 051 B: 102	FFCCFF R: 255 G: 204 B: 255	CC6699 R: 204 G: 051 B: 153	993366 R: 153 G: 051 B: 102	660033 R: 102 G: 000 B: 051	CC3399 R: 204 G: 051 B: 153	FF99CC R: 255 G: 153 B: 204	FF66CC R: 255 G: 102 B: 204	FF99FF R: 255 G: 153 B: 255	FF6699 R: 255 G: 102 B: 153	CC0066 R: 204 G: 000 B: 102
FF0066 R: 255 G: 000 B: 102	FF3399 R: 255 G: 051 B: 153	FF0099 R: 255 G: 000 B: 153	FF33CC R: 255 G: 051 B: 204	FF00CC R: 255 G: 000 B: 204	FF66FF R: 255 G: 102 B: 255	FF33FF R: 255 G: 051 B: 255	FF00FF R: 255 G: 000 B: 255	CC0099 R: 204 G: 000 B: 153	990066 R: 153 G: 000 B: 102	CC66CC R: 204 G: 102 B: 204	CC33CC R: 204 G: 051 B: 255	CC99FF R: 204 G: 153 B: 255	CC66FF R: 204 G: 102 B: 255	CC33FF R: 204 G: 051 B: 255	993399 R: 153 G: 051 B: 153
CC00CC R: 204 G: 000 B: 204	CC00FF R: 204 G: 000 B: 255	9900CC R: 153 G: 000 B: 204	990099 R: 153 G: 000 B: 153	CC99CC R: 204 G: 153 B: 204	996699 R: 153 G: 102 B: 153	663366 R: 102 G: 051 B: 102	660099 R: 102 G: 000 B: 153	9933CC R: 153 G: 051 B: 204	660066 R: 102 G: 000 B: 102	9900FF R: 153 G: 000 B: 255	9933FF R: 153 G: 051 B: 255	9966CC R: 153 G: 102 B: 204	330033 R: 051 G: 000 B: 051	663399 R: 102 G: 051 B: 153	6633CC R: 102 G: 051 B: 204
6600CC R: 102 G: 000 B: 204	9966FF R: 153 G: 255 B: 255	330066 R: 051 G: 000 B: 102	6600FF R: 102 G: 000 B: 255	6633FF R: 102 G: 051 B: 255	CC00FF R: 204 G: 000 B: 255	9999FF R: 255 G: 255 B: 255	9999CC R: 255 G: 255 B: 204	6666CC R: 102 G: 102 B: 204	6666FF R: 102 G: 102 B: 255	666699 R: 102 G: 102 B: 153	333366 R: 051 G: 051 B: 102	333399 R: 051 G: 051 B: 153	330099 R: 051 G: 000 B: 153	3300CC R: 051 G: 000 B: 204	3300FF R: 051 G: 000 B: 255
3333FF R: 051 G: 051 B: 255	3333CC R: 051 G: 051 B: 204	0066FF R: 000 G: 102 B: 255	0033FF R: 000 G: 051 B: 255	3366FF R: 051 G: 102 B: 255	3366CC R: 051 G: 102 B: 204	000066 R: 000 G: 000 B: 102	000033 R: 000 G: 000 B: 051	000099 R: 000 G: 000 B: 153	0033CC R: 000 G: 051 B: 204	0066CC R: 000 G: 102 B: 204	0033FF R: 000 G: 051 B: 255	0066FF R: 000 G: 102 B: 255	0066CC R: 000 G: 102 B: 204	99CCFF R: 153 G: 204 B: 255	6699FF R: 102 G: 153 B: 255
003366 R: 000 G: 051 B: 102	6699CC R: 102 G: 153 B: 204	006699 R: 000 G: 102 B: 153	3399CC R: 051 G: 153 B: 204	0099CC R: 000 G: 153 B: 204	66CCFF R: 102 G: 204 B: 255	3399FF R: 051 G: 153 B: 255	0033FF R: 000 G: 051 B: 255	0099FF R: 000 G: 153 B: 255	33CCFF R: 051 G: 204 B: 255	00CCFF R: 000 G: 204 B: 255	99FFFF R: 153 G: 255 B: 255	66FFFF R: 102 G: 255 B: 255	33FFFF R: 051 G: 255 B: 255	00FFFF R: 000 G: 255 B: 255	00CCCC R: 000 G: 204 B: 204
009999 R: 000 G: 153 B: 153	669999 R: 102 G: 153 B: 153	99CCCC R: 153 G: 204 B: 204	CCFFFF R: 255 G: 255 B: 255	33CCCC R: 051 G: 204 B: 204	66CCCC R: 102 G: 204 B: 204	339999 R: 051 G: 153 B: 153	336666 R: 000 G: 102 B: 102	006666 R: 000 G: 102 B: 102	003333 R: 000 G: 051 B: 051	00FFCC R: 000 G: 255 B: 204	33FFCC R: 051 G: 255 B: 204	33CC99 R: 051 G: 204 B: 153	00CC99 R: 000 G: 204 B: 153	66FFCC R: 102 G: 255 B: 204	99FFCC R: 153 G: 255 B: 204
00FF99 R: 000 G: 255 B: 153	339966 R: 051 G: 153 B: 102	006633 R: 000 G: 102 B: 051	336633 R: 051 G: 102 B: 051	669966 R: 102 G: 153 B: 102	66CC66 R: 102 G: 204 B: 102	99FF99 R: 153 G: 255 B: 153	66FF66 R: 102 G: 255 B: 102	339933 R: 051 G: 153 B: 051	99CC99 R: 153 G: 204 B: 153	66FF99 R: 102 G: 255 B: 153	33FF99 R: 051 G: 255 B: 153	33CC66 R: 051 G: 204 B: 102	00CC66 R: 000 G: 204 B: 102	66CC99 R: 102 G: 204 B: 153	009966 R: 000 G: 153 B: 102
009933 R: 000 G: 153 B: 051	33FF66 R: 051 G: 255 B: 102	00FF66 R: 000 G: 255 B: 102	CCFFCC R: 204 G: 255 B: 204	CCFF99 R: 204 G: 255 B: 153	99FF66 R: 153 G: 255 B: 102	99FF33 R: 153 G: 255 B: 051	00FF33 R: 000 G: 255 B: 051	33FF33 R: 051 G: 255 B: 051	00CC33 R: 000 G: 204 B: 051	33CC33 R: 051 G: 204 B: 051	66FF33 R: 102 G: 255 B: 051	00FF00 R: 000 G: 255 B: 000	66CC33 R: 102 G: 204 B: 051	006600 R: 000 G: 102 B: 000	003300 R: 000 G: 051 B: 000
009900 R: 000 G: 153 B: 000	33FF00 R: 051 G: 255 B: 000	66FF00 R: 102 G: 255 B: 000	99FF00 R: 153 G: 255 B: 000	66CC00 R: 102 G: 204 B: 000	00CC00 R: 000 G: 204 B: 000	33CC00 R: 051 G: 204 B: 000	339900 R: 051 G: 153 B: 000	99CC66 R: 153 G: 204 B: 102	669933 R: 102 G: 153 B: 051	99CC33 R: 153 G: 204 B: 051	336600 R: 051 G: 102 B: 000	669900 R: 102 G: 153 B: 000	99CC00 R: 153 G: 204 B: 000	CCFF66 R: 204 G: 255 B: 102	CCFF33 R: 204 G: 255 B: 051
CCFF00 R: 204 G: 255 B: 000	999900 R: 153 G: 255 B: 000	CCCC00 R: 204 G: 204 B: 000	CCCC33 R: 204 G: 204 B: 051	333300 R: 051 G: 051 B: 000	666600 R: 102 G: 102 B: 000	999933 R: 153 G: 153 B: 051	CCCC66 R: 204 G: 204 B: 102	666633 R: 102 G: 102 B: 051	999966 R: 153 G: 153 B: 102	CCCC99 R: 204 G: 204 B: 153	FFFFCC R: 255 G: 255 B: 204	FFFF99 R: 255 G: 255 B: 153	FFFF66 R: 255 G: 255 B: 102	FFFF33 R: 255 G: 255 B: 051	FFFF00 R: 255 G: 255 B: 000
FFCC00 R: 255 G: 204 B: 000	FFCC66 R: 255 G: 204 B: 102	FFCC33 R: 204 G: 204 B: 051	CC9933 R: 204 G: 153 B: 051	996600 R: 153 G: 102 B: 000	CC9900 R: 204 G: 153 B: 000	FF9900 R: 255 G: 153 B: 000	CC6600 R: 204 G: 102 B: 000	993300 R: 153 G: 051 B: 000	CC6633 R: 204 G: 102 B: 051	663300 R: 102 G: 051 B: 000	FF9966 R: 255 G: 153 B: 102	FF6633 R: 255 G: 102 B: 051	FF9933 R: 255 G: 153 B: 051	FF6600 R: 255 G: 102 B: 000	CC3300 R: 204 G: 051 B: 000
996633 R: 153 G: 102 B: 051	330000 R: 051 G: 000 B: 000	663333 R: 102 G: 051 B: 051	996666 R: 153 G: 102 B: 102	CC9999 R: 204 G: 153 B: 153	993333 R: 153 G: 051 B: 102	CC6666 R: 204 G: 153 B: 102	FFCCCC R: 255 G: 204 B: 204	FF3333 R: 255 G: 051 B: 051	CC3333 R: 204 G: 051 B: 051	FF6666 R: 255 G: 102 B: 102	660000 R: 102 G: 000 B: 000	990000 R: 153 G: 000 B: 000	CC0000 R: 204 G: 000 B: 000	FF0000 R: 255 G: 000 B: 000	FF3300 R: 255 G: 051 B: 000
CC9966 R: 204 G: 153 B: 102	FFCC99 R: 255 G: 204 B: 153	FFFFFF G: 255 B: 255	CCCCCC R: 204 G: 204 B: 204	999999 R: 153 G: 153 B: 153	666666 R: 102 G: 102 B: 102	333333 R: 051 G: 051 B: 051	000000 R: 000 G: 000 B: 000								

Reprezentare imagine: matrice bidimensională de triplete sau vector de 3 matrice bidimensionale

Tipul abstract de dată (TAD) Listă



Liste de activități



O listă stocată în memorie

- De ce organizăm astfel informațiile?
- Ce operații vom executa?

TAD Listă

Definiție:

O listă este o structură de date **omogenă**, secvențială formată din elemente (noduri) între care există cel puțin o relație de ordine.

Un element al unei liste are forma generală:



Un element al listei se identifică prin succesorul sau predecesorul său.

Operații cu liste

1. Operații de caracterizare

- ✓ Determinarea numărului de elemente
- ✓ Localizarea unui element cu anumite proprietăți (cautare)
- ✓ Parcurgerea (afișarea) listei

2. Operații care modifică conținutul listei

- ✓ Initializarea
- ✓ Inserarea unui element în oricare loc din listă
- ✓ Ștergerea unui element din listă
- ✓ Modificarea conținutului unui element
- ✓ Operații complexe (ex. Sortarea listei după câmpul de date)

3. Alte operații complexe

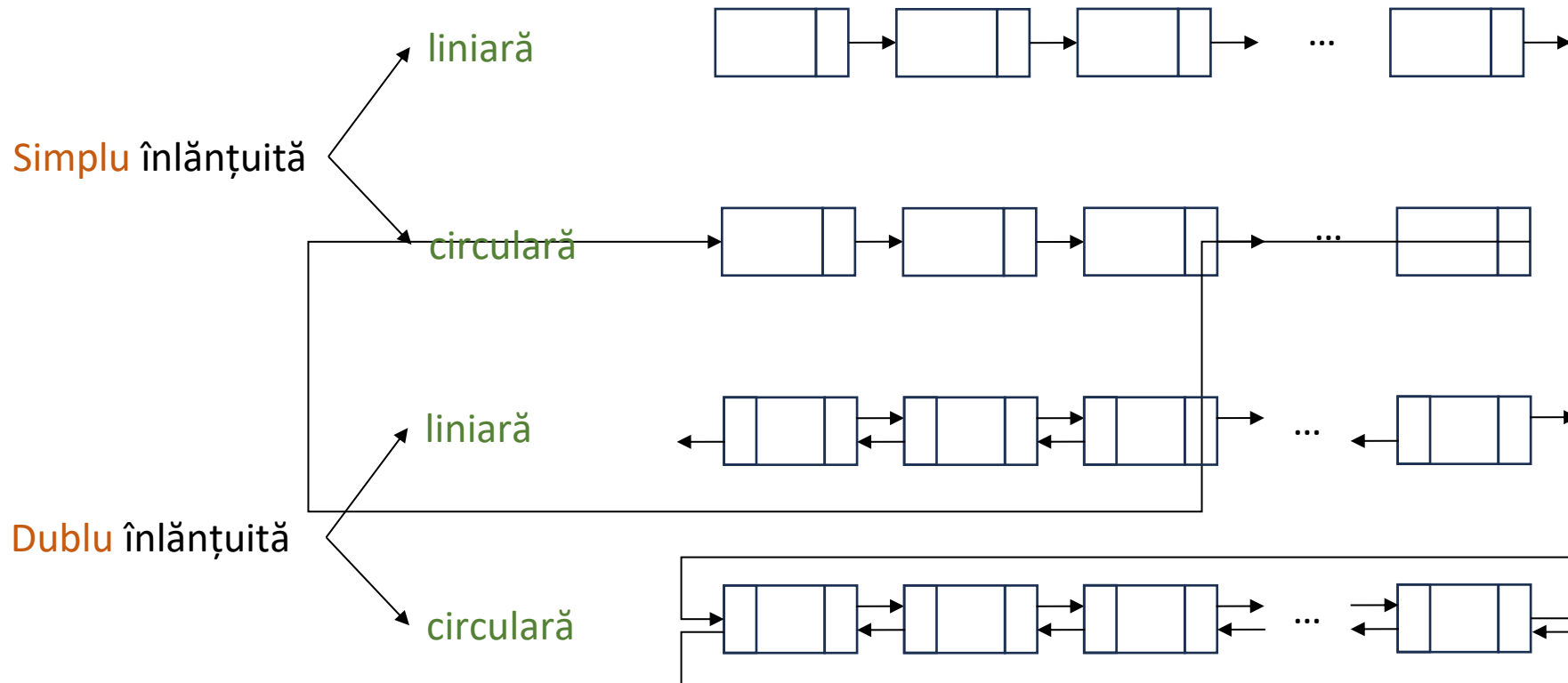
- ✓ Selecția elementelor după un criteriu și crearea unei noi liste
- ✓ Separea unei liste în mai multe liste după un anumit criteriu
- ✓ Combinarea unor liste prin concatenare sau interclasare

TAD Listă

Un element al unei liste are forma generală:



Tipuri de liste:



TAD Listă

Un element al unei liste are forma generală:



Implementare:

- Dinamică
- Statică

TAD Listă

Un element al unei liste are forma generală:



Lista liniară simplu înlanțuită:



Implementare statică

	<i>cap</i>						
	0	1	2	3	4	5	6
<i>data</i>	7	2	29	32	14	10	5
<i>succesor</i>	6	5	4	1	-1	0	2
	} <i>vec</i>						

```
struct Element{  
    Atom data;  
    uint succesor;  
};
```

```
struct Lista{  
    Element vec[7];  
    uint cap;  
};
```

```
Lista myList;  
myList.cap = 3;  
myList.vec[myList.cap].data = 32;  
myList.vec[myList.cap].succesor = 1;  
....
```

TAD Listă

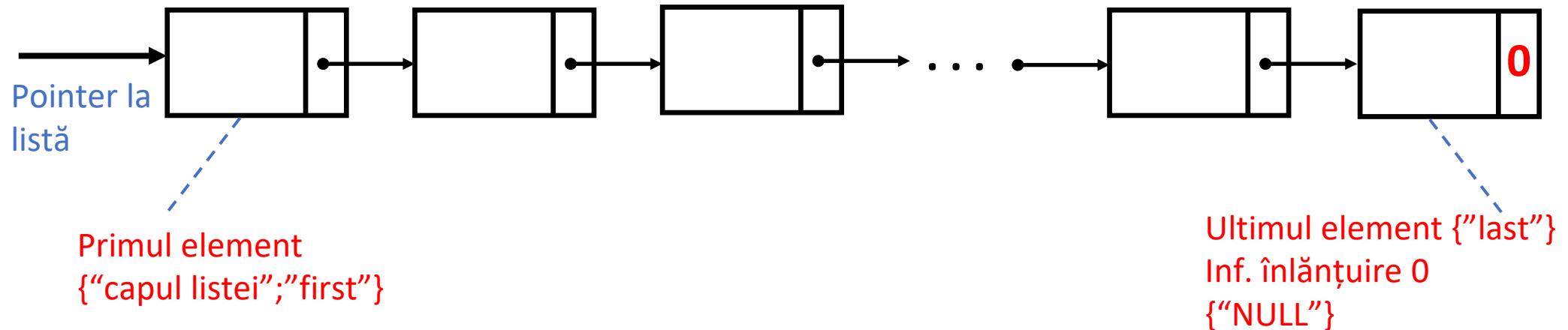
Un element al unei liste are forma generală:



Lista liniară simplu înlanțuită:

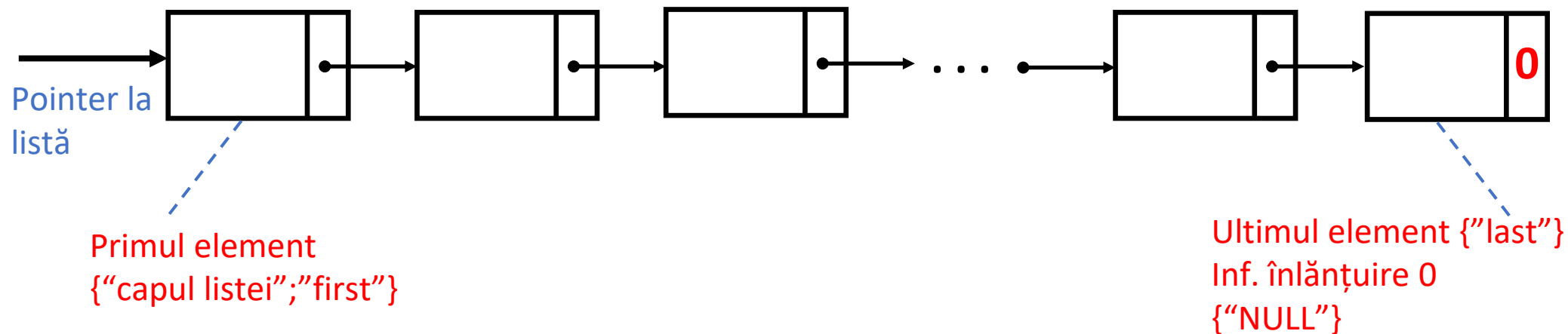


Implementare **dinamică**



Listă lineară simplu înlănțuită – Implementare **dinamică**

- Un element (nod) din listă este de un tip structură și are (cel puțin) două câmpuri: un câmp de date (sau mai multe) și un câmp de legătură care este pointer la succesori.
- Sunt două elemente particulare:
 - ✓ **primul element** (“capul listei”) - singurul prin care se face accesul la listă;
 - ✓ **ultimul element** – al cărui câmp de legătură este NULL.
- Parcurgerea se face într-un singur sens, pornind de la primul element
- Dimensiunea listei (numărul de elemente) este teoretic nelimitată dar practic este limitată de memoria de lucru disponibilă



Lista liniară simplu înlănțuită – Implementare **dinamică**

Declarații

```
typedef int Atom; // orice tip predefinit
```

```
struct Element  
{  
    Atom data;  
    Element *succ;  
};
```

... main

```
Element *cap; // pointer la lista ("cap")
```

Notatii pseudocod

p – adresa unui element din listă {**Element** $*p$ }

$data(p)$ – informație valoare (Atom) { $p \rightarrow data$ }

$succ(p)$ – informație înlănțuire (adresa următorului element) { $p \rightarrow succ$ }

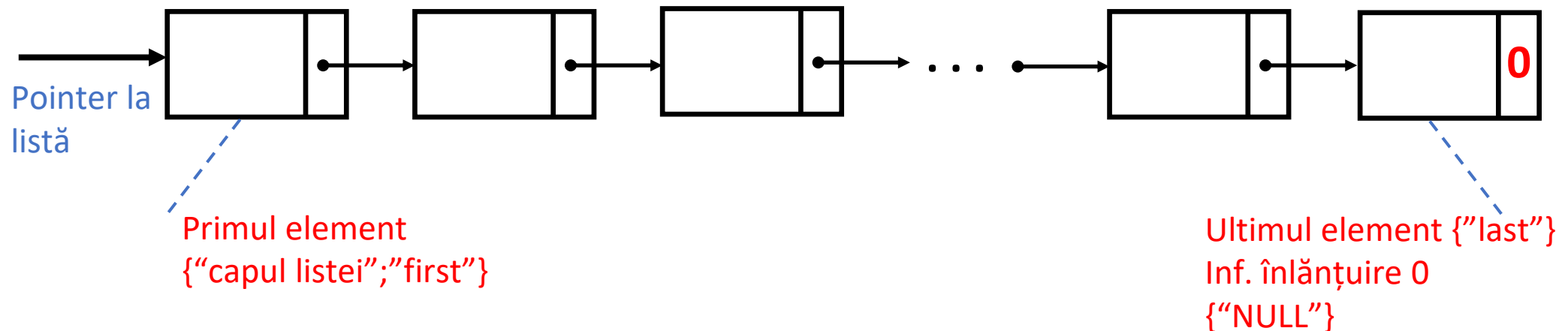
$p \leftarrow get_sp()$ – alocarea unei zone de memorie pentru un element { $p = new\ Element$ }

$free_sp(p)$ – eliberarea zonei de memorie ocupate de elementul de pointer p {**delete** p }

Operații cu liste liniare simplu înlănțuite

Parcurgerea listei

```
P ← cap; // pointer la primul element  
While p ≠ 0 Do  
    prelucrare (data(p));  
    p ← succ(p); // pointer la următorul element  
EndWhile
```

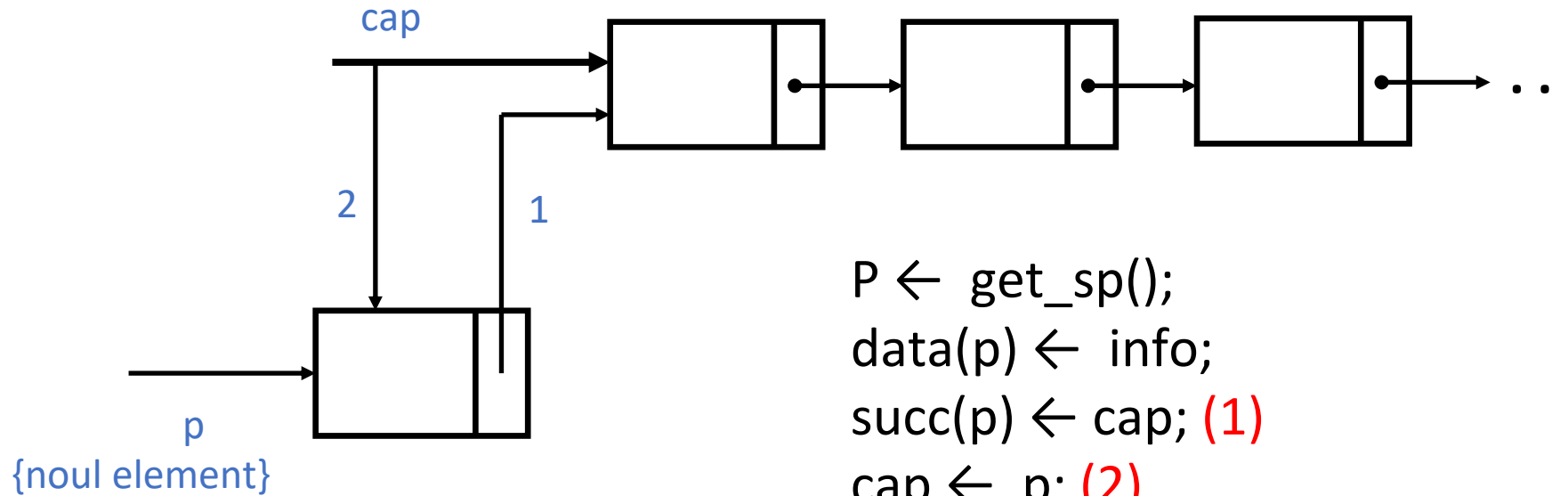


Operații cu liste liniare simplu înlănțuite

Inserarea unui element

Indiferent de locul inserării este necesară alocarea memoriei pentru noul element!

Inserarea în fața primului element



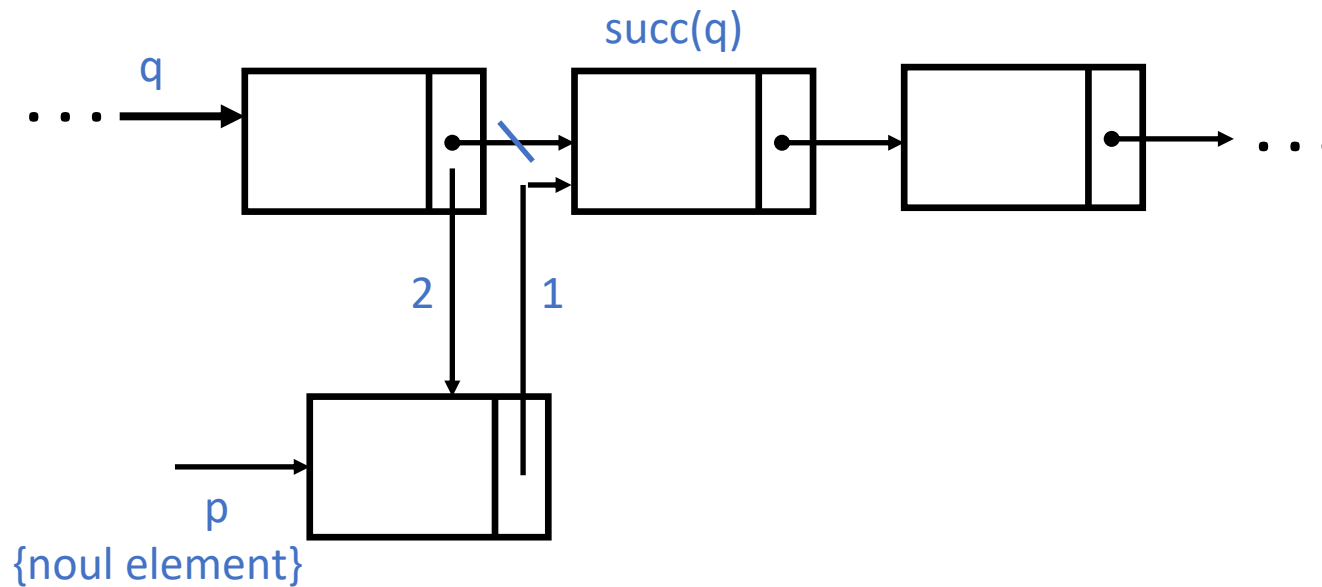
```
P ← get_sp();  
data(p) ← info;  
succ(p) ← cap; (1)  
cap ← p; (2)
```

Complexitate $O(1)$

Operații cu liste liniare simplu înlănțuite

Inserarea unui element în interiorul listei

Se parcurge lista până la elementul de pointer q, după care se face inserarea!



```
P ← get_sp();  
data(p) ← info;  
succ(p) ← succ (q); (1)  
succ(q) ← p; (2)
```

Complexitate $O(n)$

Secvența este valabilă și pentru inserarea după ultimul element.

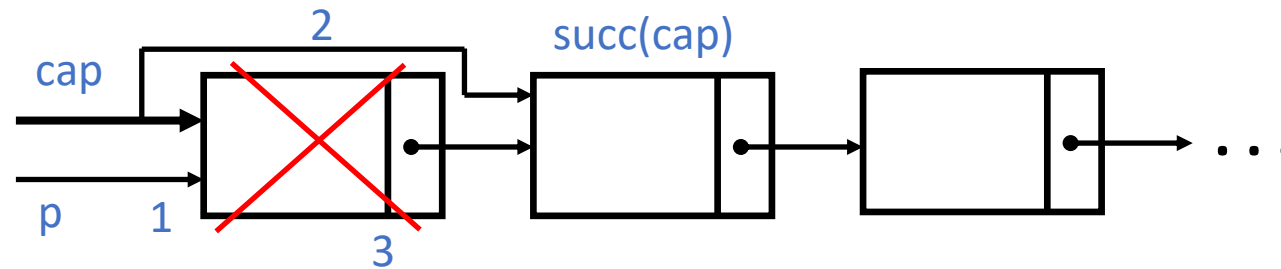
Se poate insera un nou nod în fața nodului de pointer q?

Operații cu liste liniare simplu înlănțuite

Stergerea unui element

Operația de ștergere implică refacerea legăturilor după detașarea nodului din listă și eliberarea zonei de memorie.

Ștergerea primului element



$P \leftarrow \text{cap};$ (1)

$\text{cap} \leftarrow \text{succ}(\text{cap});$ (2)

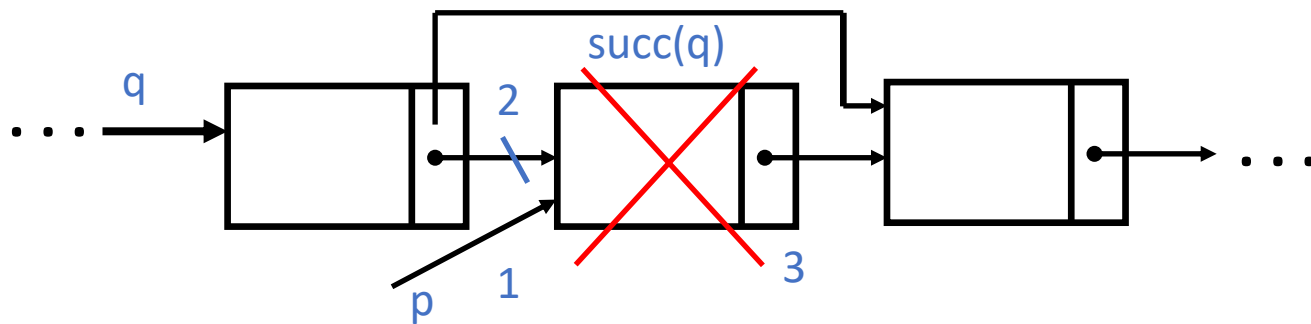
$\text{free_sp}(p);$ (3)

Complexitate $O(1)$

Operații cu liste liniare simplu înlănțuite

Ștergerea unui element din interiorul listei

Se parcurge lista până la elementul de pointer q, după care se face ștergerea!



```
p ← succ(q); (1)  
succ(q) ← succ(p); (2)  
free_sp(p); (3)
```

Complexitate $O(n)$

Secvența este valabilă și pentru ștergerea ultimului element.

Se poate șterge un nod de pointer q?

Exemplu: crearea unei liste liniare simplu inlantuite prin inserari in fata

```
struct Nod{
    int data;
    Nod * succ;
};
```

Inserare in fata primului element

```
void Insert(Nod *&cap, int val)
{
    //transfer prin referinta
    Nod *p;
    p=new Nod;
    p->data=val;
    p->succ=cap;
    cap=p;
}
```

Creare lista

```
void CreateList(Nod*& cap)
{
    //transfer prin referinta
    int n;
    cout << "introduceti valori (0- termina)!"
    << endl;
    cin >> n;
    while (n)
    {
        Insert(cap, n);
        cin >> n;
    }
}
```

Apel in main()

```
Nod *p=0; //initializarea!!
CreateList(p);
```

Exemplu: functie care afiseaza **valoarea elementului din mijlocul listei** fara sa numere elementele acesteia

Se considera doi pointeri pentru parcurgerea listei, dintre care unul se “deplaseaza” de doua ori mai repede. Initial ambii indica primul element. Cand pointerul “mai rapid” ajunge la sfarsitul listei, cel mai “lent” va fi pe nodul din mijloc.

```

Middle(cap)
    p←cap;
    q←cap;
    While(q≠0 and succ(q)≠0) Do
        p←succ(p);
        q←succ(succ(q));
    EndWhile
    Print data(p);
End
```

Discutie: ce se afiseaza daca lista
are numar par de noduri?