

Structuri de date – Curs 3

Prof. univ. dr. Cristian CIUREA
Departamentul de Informatică și Cibernetică Economică
Academia de Studii Economice din București
cristian.ciurea@ie.ase.ro

Agenda

- ▶ Structura de tip stivă
- ▶ Caracteristicile structurii de tip stivă
- ▶ Structura de tip coadă
- ▶ Caracteristicile structurii de tip coadă
- ▶ Aplicabilitate stive și cozi

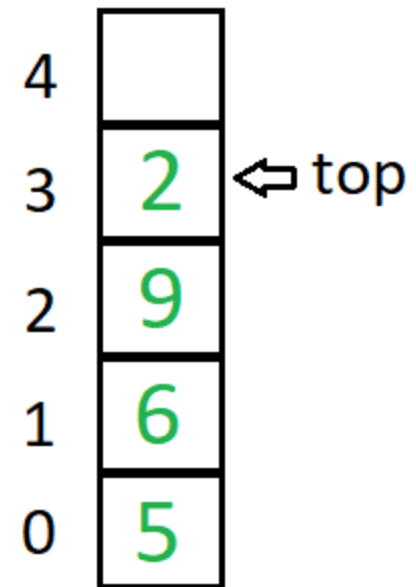
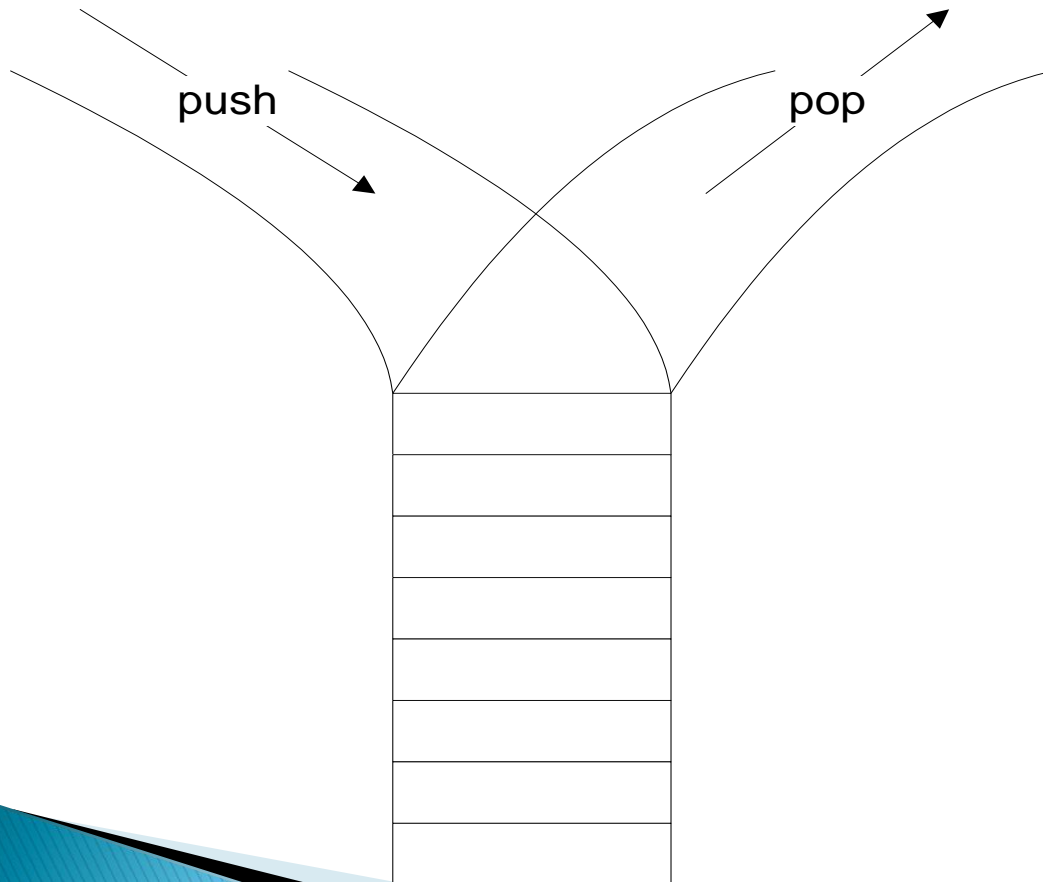
Stiva

Structura de tip stivă:

- ▶ caz particular al **listei liniare simple**;
- ▶ structură de date logică: implementarea este realizată utilizând alte structuri de date;
- ▶ structură de date omogenă: **toate elementele sunt de același tip**;
- ▶ **două operații de bază**: adăugarea și extragerea unui element;
- ▶ disciplina de acces: ***LIFO (Last In First Out)*** – toate inserările (**push**) și extragerile (**pop**) sunt făcute la unul din capetele structurii de implementare, denumit **vârful stivei**.

Stiva

► mecanismul de stivă:



Stack

Stiva

Caracteristicile structurii de tip stivă:

- ▶ vârful stivei poate fi citit, șters sau în fața lui se poate insera un nou nod care devine cap de stivă;
- ▶ stiva poate fi implementată ca o listă liniară simplă pentru care operațiile de acces (inserare, ștergere, accesare element) sunt restricționate astfel:
 - inserarea se face doar în fața primului element al listei, capul listei;
 - accesarea, respectiv ștergerea acționează doar asupra primului element al listei.

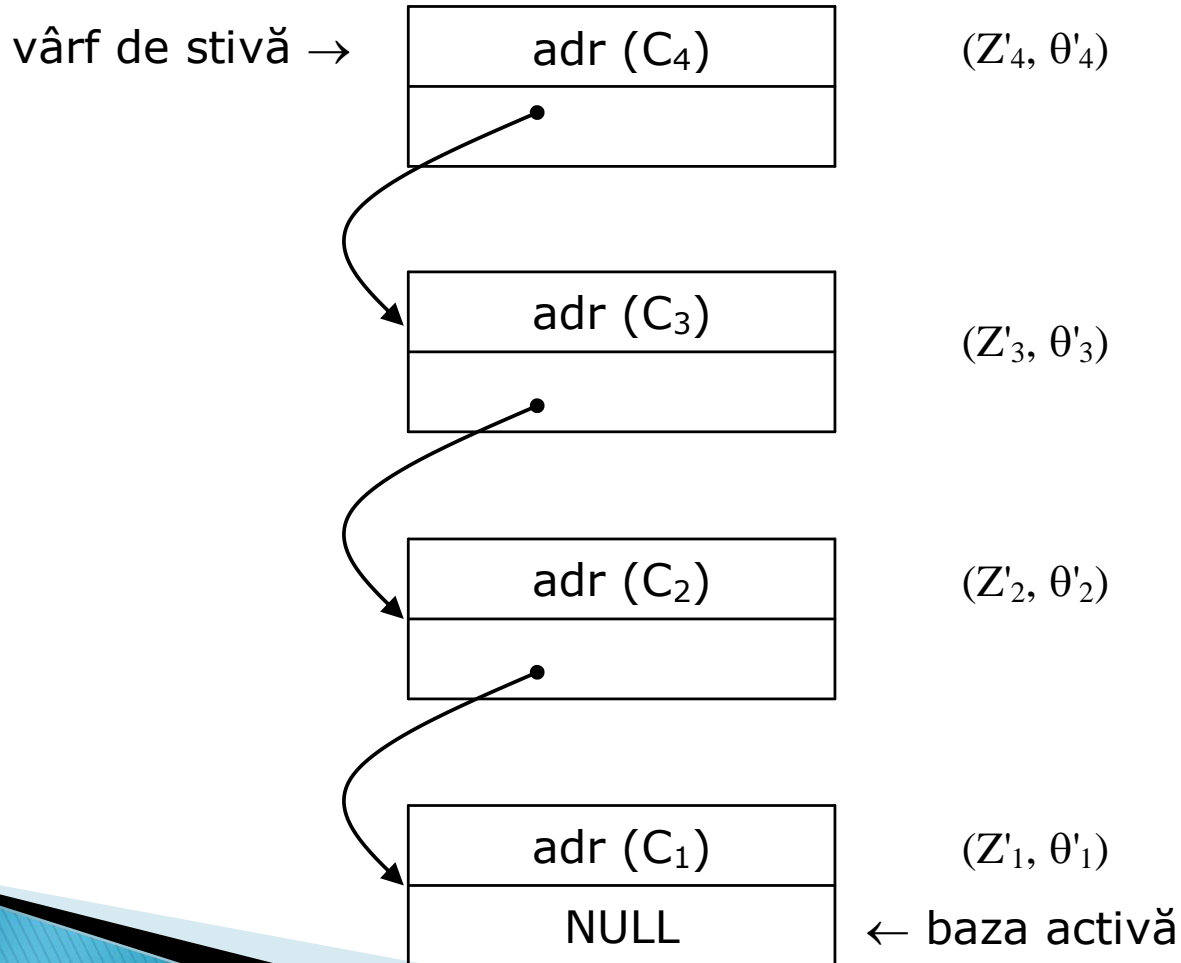
Stiva

Caracteristicile structurii de tip stivă:

- ▶ în timp ce $cont(\theta_n) = NULL$ în cazul listei, $cont(\theta'_1) = NULL$ în cazul stivei;
- ▶ în timp ce $succ(Z_j) = Z_{j+1}$ în cazul listei, în cazul stivei $succ(Z'_j) = Z'_{j-1}$;
- ▶ în timp ce $pred(Z_j) = Z_{j-1}$ în cazul listei, în cazul stivei $pred(Z'_j) = Z'_{j+1}$;
- ▶ în timp ce parcurgerea în cazul listei este de la (Z_1, θ_1) spre (Z_n, θ_n) , în cazul stivei, parcurgerea este de la (Z'_n, θ'_n) spre (Z'_1, θ'_1) .

Stiva


► structura unei stive cu 4 noduri:



Stiva

- ▶ exemplu definire nod stiva:

```
struct nod_stiva  
{  
    book inf;  
    nod_stiva *nextstack;  
};
```



```
struct book  
{  
    int ISBN;  
    float price;  
};
```


Stiva

- ▶ exemplu inserare valori (1, 20) si (2, 30):

Name	Value	Type
&stiva	0x0024fadc	nod_stiva **
	0x00901540 {inf={...} nextstack=0x009014f8 }	nod_stiva *
inf	{ISBN=2 price=30.000000 }	book
ISBN	2	int
price	30.000000	float
nextstack	0x009014f8 {inf={...} nextstack=0x00000000 }	nod_stiva *
inf	{ISBN=1 price=20.000000 }	book
ISBN	1	int
price	20.000000	float
nextstack	0x00000000 {inf={...} nextstack=??? }	nod_stiva *
inf	{ISBN=??? price=??? }	book
ISBN	CXX0030: Error: expression cannot be evaluated	
price	CXX0030: Error: expression cannot be evaluated	
nextstack	CXX0030: Error: expression cannot be evaluated	
carte	{ISBN=2 price=30.000000 }	book
ISBN	2	int
price	30.000000	float

Stiva

```
struct carte
```

```
{
```

```
    int cod;
```

```
    char* titlu;
```

```
    float pret;
```

```
};
```

```
struct nodStiva
```

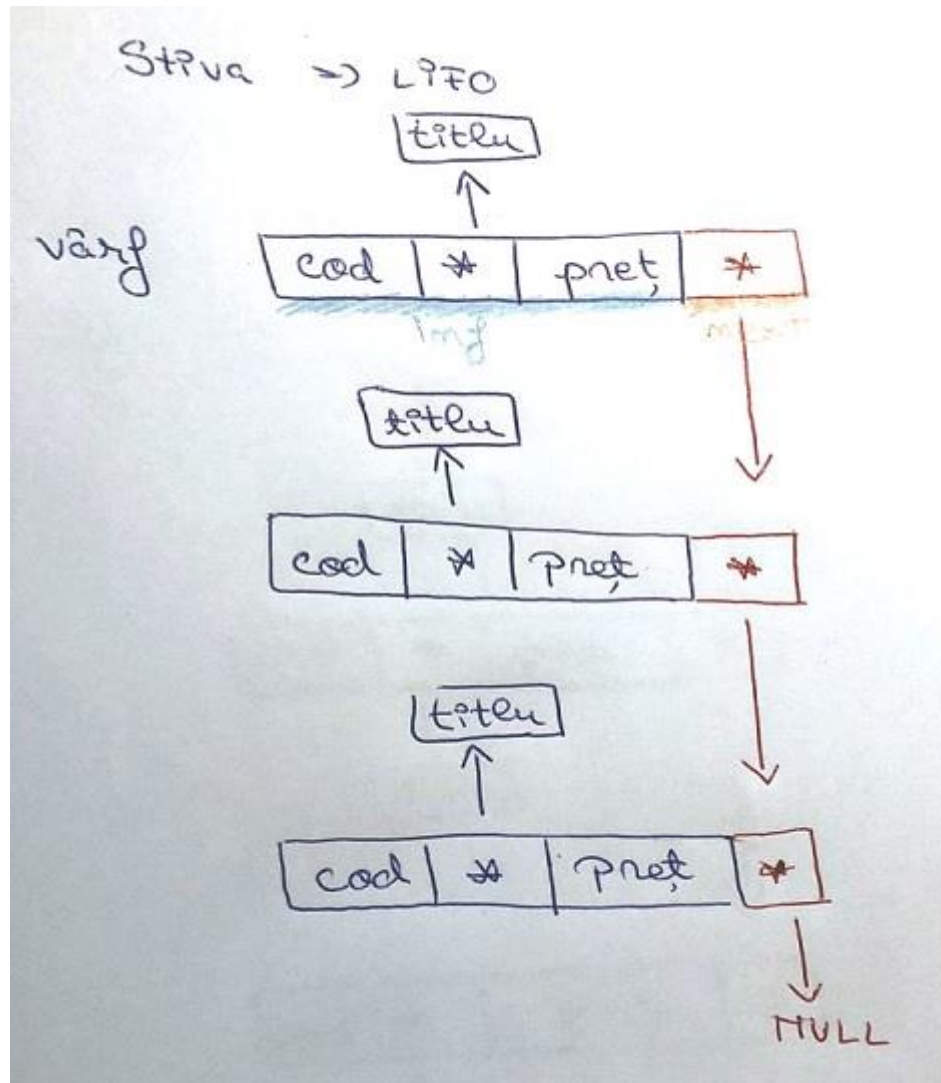
```
{
```

```
    carte inf;
```

```
    nodStiva* next;
```

```
};
```

Stiva



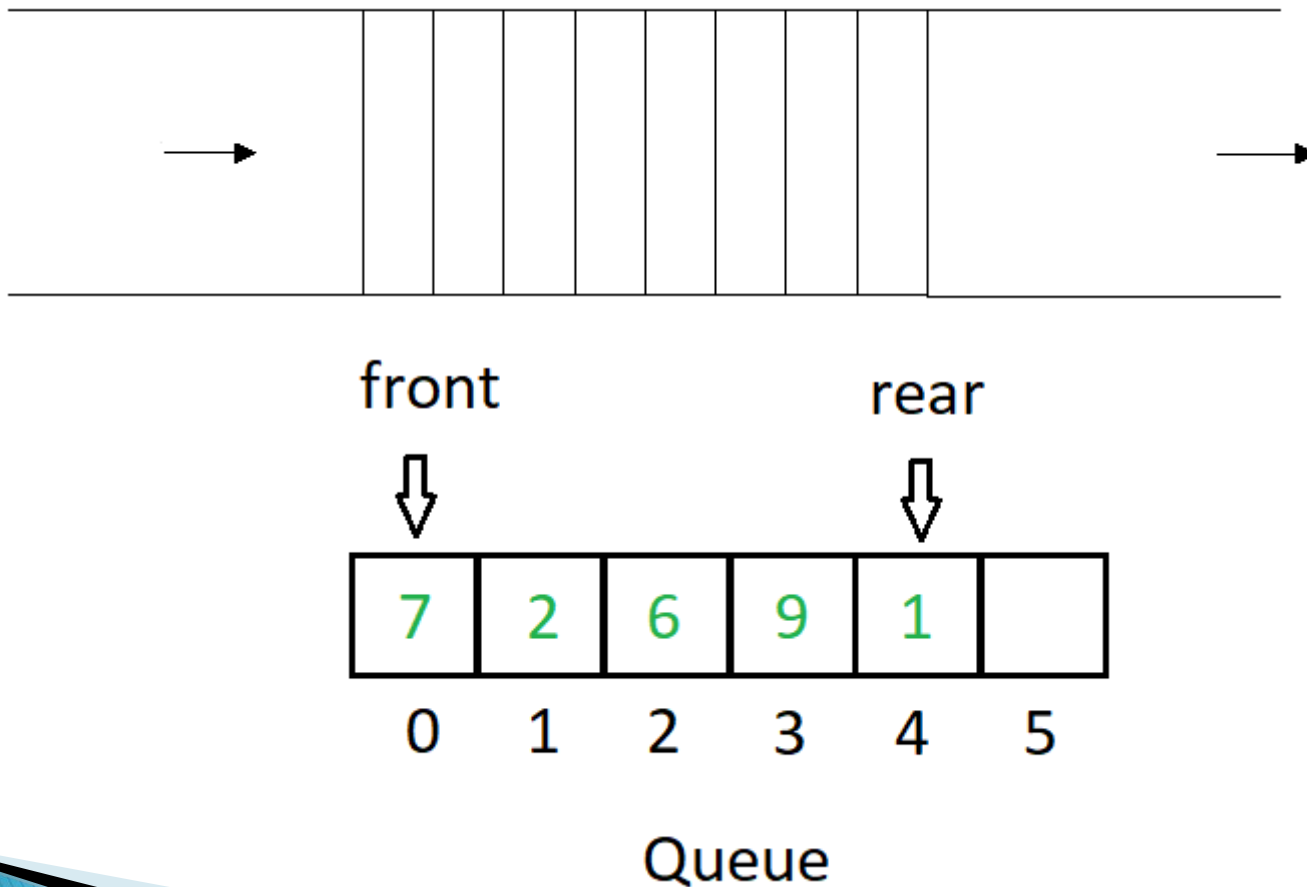
Coada

Structura de tip coadă:

- ▶ caz particular al **listei liniare simple**;
- ▶ structură de date logică: implementarea este realizată utilizând alte structuri de date;
- ▶ structură de date omogenă: **toate elementele sunt de același tip**;
- ▶ **două operații de bază**: adăugarea și extragerea unui element;
- ▶ disciplina de acces: ***FIFO (First In First Out)*** – toate inserările (**put**) se fac la un capăt (sfârșitul cozii) și extragerile (**get**) sunt făcute la celălalt capăt (începutul cozii).

Coada

- ▶ mecanismul de coadă:



Coada

Caracteristicile structurii de tip coadă:

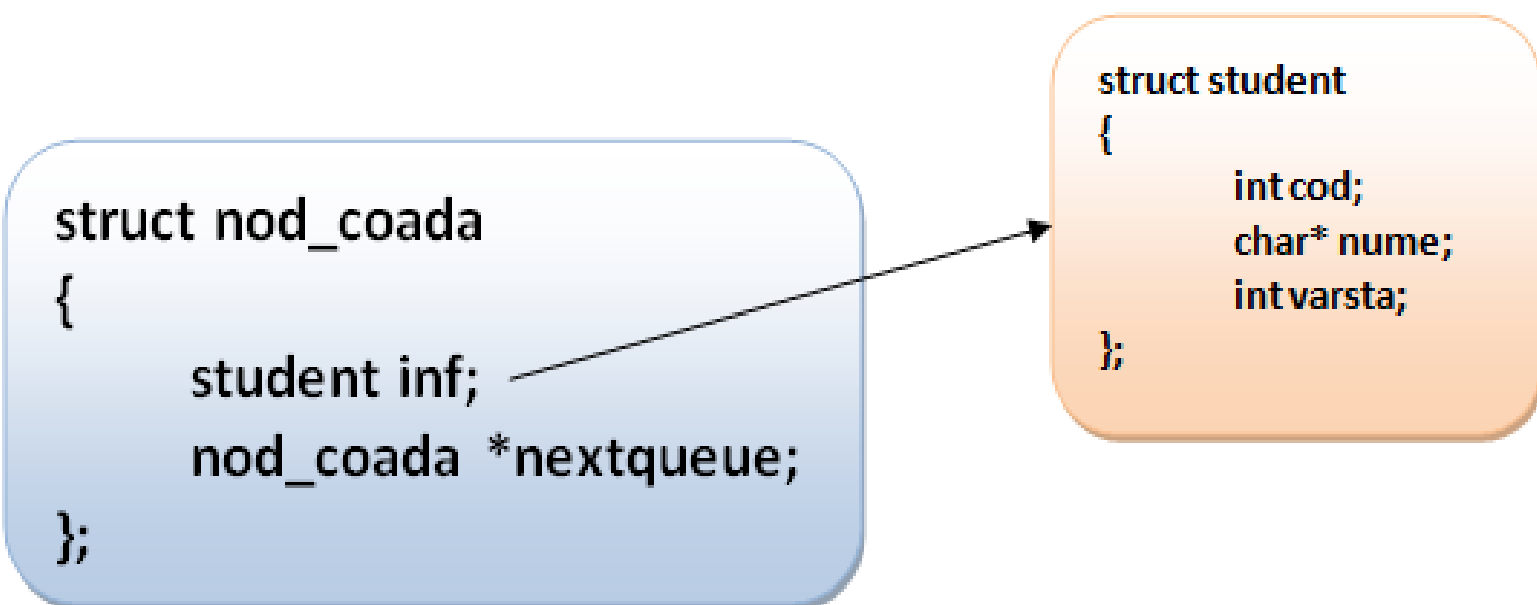
- ▶ coada poate fi implementată folosind o listă liniară simplu înlănțuită în care operațiile de acces sunt restricționate astfel:
 - adăugarea se face tot timpul la începutul listei liniare simple, iar extragerea se face de la sfârșitul listei, sau
 - adăugarea se face la sfârșitul listei, iar extragerea se face din capul listei.

Coada

- ▶ exemplu definire nod coadă:

```
struct nod_coada
{
    student inf;
    nod_coada *nextqueue;
};
```

```
struct student
{
    int cod;
    char* nume;
    int varsta;
};
```



Coadă

- ▶ exemplu inserare valori (1, Gigel, 21) si (2, Maria, 23):

Name	Value	Type
&p	0x0026fd38	nod_coadă **
	0x00341540 {inf={...} nextqueue=0x00341620 }	nod_coadă *
inf	{cod=1 nume=0x00341590 "Gigel" varsta=21 }	student
cod	1	int
nume	0x00341590 "Gigel"	char *
varsta	21	int
nextqueue	0x00341620 {inf={...} nextqueue=0x00000000 }	nod_coadă *
inf	{cod=2 nume=0x00341670 "Maria" varsta=23 }	student
cod	2	int
nume	0x00341670 "Maria"	char *
varsta	23	int
nextqueue	0x00000000 {inf={...} nextqueue=??? }	nod_coadă *
&u	0x0026fd2c	nod_coadă **
	0x00341620 {inf={...} nextqueue=0x00000000 }	nod_coadă *
inf	{cod=2 nume=0x00341670 "Maria" varsta=23 }	student
cod	2	int
nume	0x00341670 "Maria"	char *
varsta	23	int
nextqueue	0x00000000 {inf={...} nextqueue=??? }	nod_coadă *
stud	{cod=2 nume=0x003415d8 "Maria" varsta=23 }	student

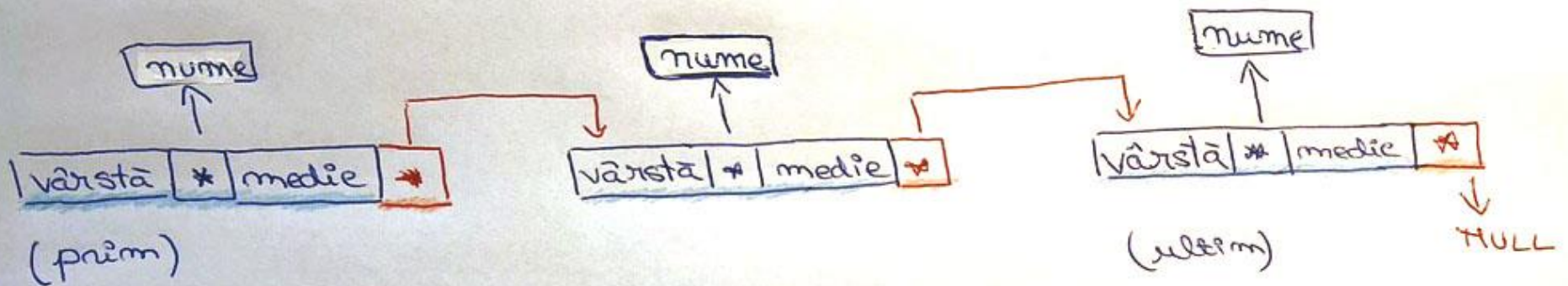
Coadă

```
struct student
{
    int varsta;
    char* nume;
    float medie;
};

struct nodCoadă
{
    student inf;
    nodCoadă* next;
};
```

Coadă

Coadă \Rightarrow FIFO



Stive si cozi

- ▶ Zone de aplicabilitate ale structurii de tip stivă:
 - implementarea algoritmilor recursivi (quicksort);
 - traversarea structurilor arborescente (grafuri);
 - evaluarea expresiilor matematice;
 - conversia între baze de numerație;
 - verificarea unui șir dacă este **palindrom**;

Able was I ere I saw Elba
Was it a cat I saw
Eva, can I see bees in a cave
No lemon, no melon

Stive si cozi

- ▶ Zone de aplicabilitate ale structurii de tip **stivă**:
 - Operațiunea de anulare a modificărilor (undo) se efectuează prin stive.
 - Stiva este folosită în multe mașini virtuale (JVM).
 - Navigare înainte – înapoi în browser
 - Istoricul site-urilor web vizitate
 - Jurnalele de mesaje și toate mesajele primite sunt aranjate pe bază de stivă
 - Jurnalele de apeluri, e-mailuri, orice galerie de fotografii Google, descărcări YouTube, notificări (ultima recepționată apare prima afișată)
 - etc.

Stive si cozi

- ▶ Zone de aplicabilitate ale structurii de tip **coadă**:
 - implementare cozi de priorități;
 - simulare procese de servire;
 - traversarea structurilor arborescente (grafuri);
 - Sistemul de operare folosește coada pentru programarea job-urilor.
 - Pachetele de date în comunicații sunt aranjate sub formă de coadă.

Stive si cozi

- ▶ Zone de aplicabilitate ale structurii de tip coadă:
 - Trimiterea unui e-mail presupune să fie pus în coada de procesare.
 - Încărcarea și descărcarea fotografiilor într-un/dintr-un album.
 - Majoritatea solicitărilor și proceselor de pe internet utilizează coada.
 - Comutarea între mai multe aplicații Windows utilizează coada.
 - Implementare buffer de mesaje între procese/sisteme sub formă de coadă.
 - etc.

Stive si cozi

Evaluarea expresiilor matematice ce utilizează ca structură de date principală stiva:

- rearanjarea expresiei într-o anumită formă astfel încât ordinea operațiilor să fie clară și evaluarea să necesite o singură parcurgere a expresiei;
- forma poloneză: matematicianul de origine poloneză Jan Lukasiewicz;
- forma poloneză: scrierea operatorilor înaintea operanzilor;
- forma poloneză inversă: operatorii sunt scriși în urma operanzilor.

Stive si cozi

Avantajele formeii poloneze inverse (scriere postfixată) față de scrierea prefixată (forma poloneză) sau infixată (expresia matematică):

- ordinea în care se efectuează operațiile este clară;
- parantezele nu mai sunt necesare;
- evaluările sunt ușor de efectuat cu ajutorul calculatorului.

Stive si cozi

Un algoritm de transformare din expresie matematică în scriere postfixată: Edsger Dijkstra (algoritmul macazului – Dijkstra Shunting Algorithm):

- utilizare stivă în care sunt păstrați operatorii și din care sunt eliminați și transferați în scrierea postfixată;
- fiecare operator are atribuită o ierarhie.

Stive si cozi

- ▶ ierarhia operatorilor:

Operator	Ierarhie
([{	1
)] }	2
+ -	3
* /	4

Stive si cozi

- ▶ forme ale scrierii unei expresii matematice:

Expresia matematică (scriere infixată)	Expresia în forma poloneză (scriere prefixată)	Expresia în forma poloneză inversă (scriere postfixată)
$4 + 5$	$+ 4 5$	$4 5 +$
$4 + 5 * 5$	$+ 4 * 5 5$	$4 5 5 * +$
$4 * 2 + 3$	$+ * 4 2 3$	$4 2 * 3 +$
$4 + 2 + 3$	$+ + 4 2 3$	$4 2 + 3 +$
$4 * (2 + 3)$	$* 4 + 2 3$	$4 2 3 + *$

Stive si cozi

Algoritmul de transformare este:

- ▶ se inițializează stiva și scrierea postfixată;
- ▶ atât timp cât nu s-a ajuns la sfârșitul expresiei matematice:
 - se citește următorul element din expresie;
 - dacă este valoare se adaugă în scrierea postfixată;
 - dacă este „(” se introduce în stivă;
 - dacă este „)” se transferă elemente din stivă în scrierea postfixată până la „(”;
- ▶ altfel:
 - atât timp cât ierarhia operatorului din vârful stivei este mai mare ca ierarhia operatorului curent, se trece elementul din vârful stivei în scrierea postfixată;
 - se introduce operatorul curent în stivă;
 - se trec toți operatorii rămași pe stivă în scrierea postfixată.

Stive si cozi

Algoritmul de evaluare este:

- ▶ se inițializează stiva;
- ▶ atât timp cât nu s-a ajuns la sfârșitul scrierii postfixate:
 - se citește următorul element;
 - dacă este valoare se depune pe stivă;
 - altfel (este operator):
 - se extrage din stivă elementul y ;
 - se extrage din stivă elementul x ;
 - se efectuează operația x operator y ;
 - se depune rezultatul pe stivă;
- ▶ ultima valoare care se află pe stivă este rezultatul expresiei.

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack:

Output:2

Stive si cozi – Infix to Postfix

$$2 * (4 + 3) + 9 / 3$$

Stack: *

Output: 2

Stive si cozi – Infix to Postfix

$2*(4+3)+9/3$

Stack:*(

Output:2

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack:*(

Output:24

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack:*(+

Output:24

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack:*(+

Output:243

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack:*

Output:243+

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack: +

Output: 243+*

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack: +

Output: 243+*9

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack: + /

Output: 243+*9

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack: + /

Output: 243+*93

Stive si cozi – Infix to Postfix

$$2*(4+3)+9/3$$

Stack:

Output: 2 4 3 + * 9 3 / +

Bibliografie

- ▶ Ion Ivan, Marius Popa, Paul Pocatilu (coordonatori) – *Structuri de date*, Editura ASE, București, 2008.
 - Cap. 11. Stive și cozi
- ▶ <https://www.geeksforgeeks.org/real-time-application-of-data-structures>
- ▶ <https://acs.ase.ro/data-structures>