Cursul nr. 5

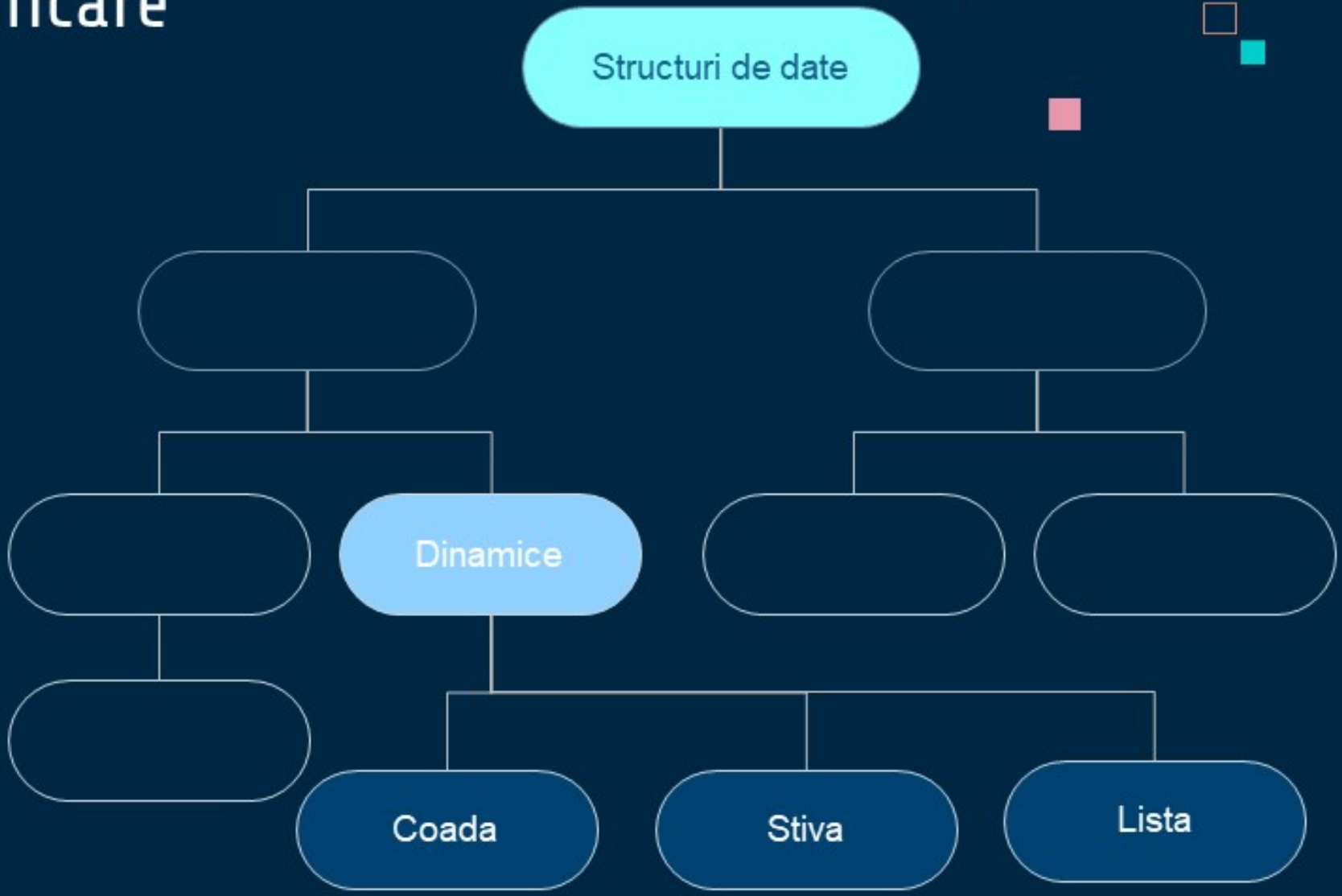
STRUCTURI DE DATE

dinamice

Liste particulare: stive, cozi

Lector dr. Dorin IORDACHE

Clasificare



Agenda

01

Stiva

02

Coada

Stiva

01

Stiva

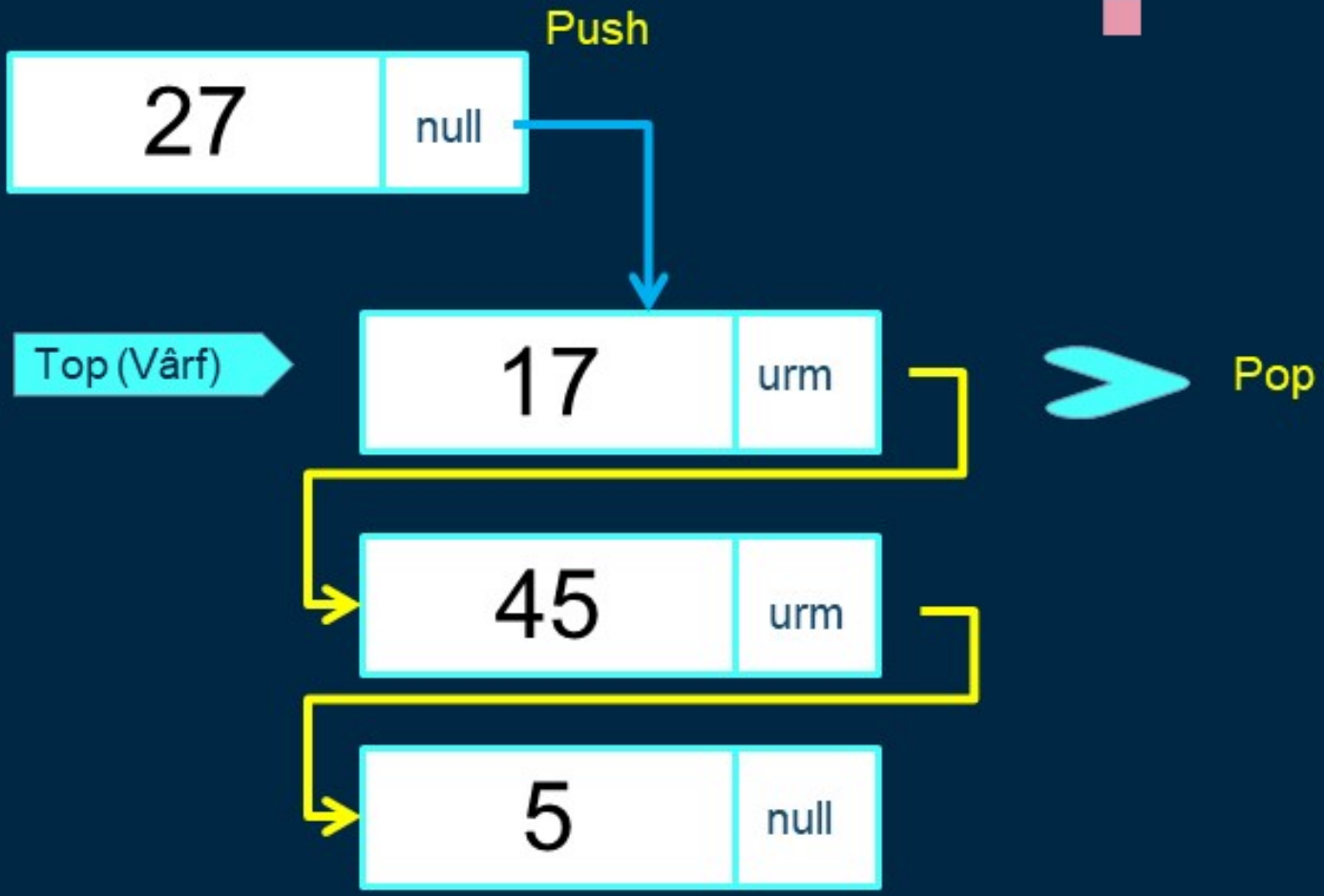
Este o structură de date liniară care urmează o anumită ordine în care sunt efectuate operațiunile.

LIFO (Ultimul intrat, primul ieşit):

elementul care este introdus ultimul va ieși primul.

Exemplu:
Un teanc de farfurii ținute una peste.
Ultima farfurie este deasupra si din moment ce scoatem farfuria care este deasupra, putem spune ca:
farfuria care a fost pusa ultima iese prima.

Stiva

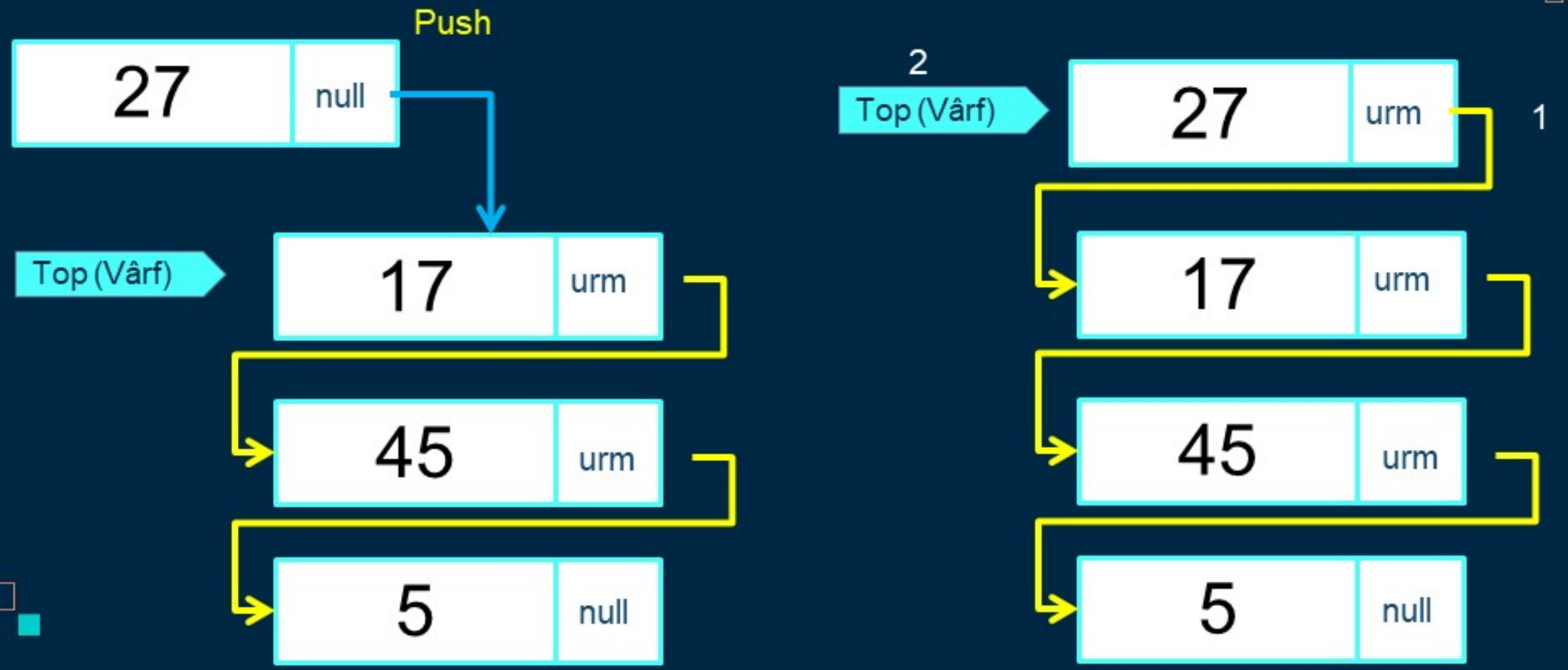


Operatii

- push() - a introduce un element în stivă
- pop() - a elimina un element din stivă
- top() - returnează elementul din vârful stivei.
- isEmpty() - returnează adevărat dacă stiva este goală, altfel false
- size() - returnează dimensiunea stivei

Operatii

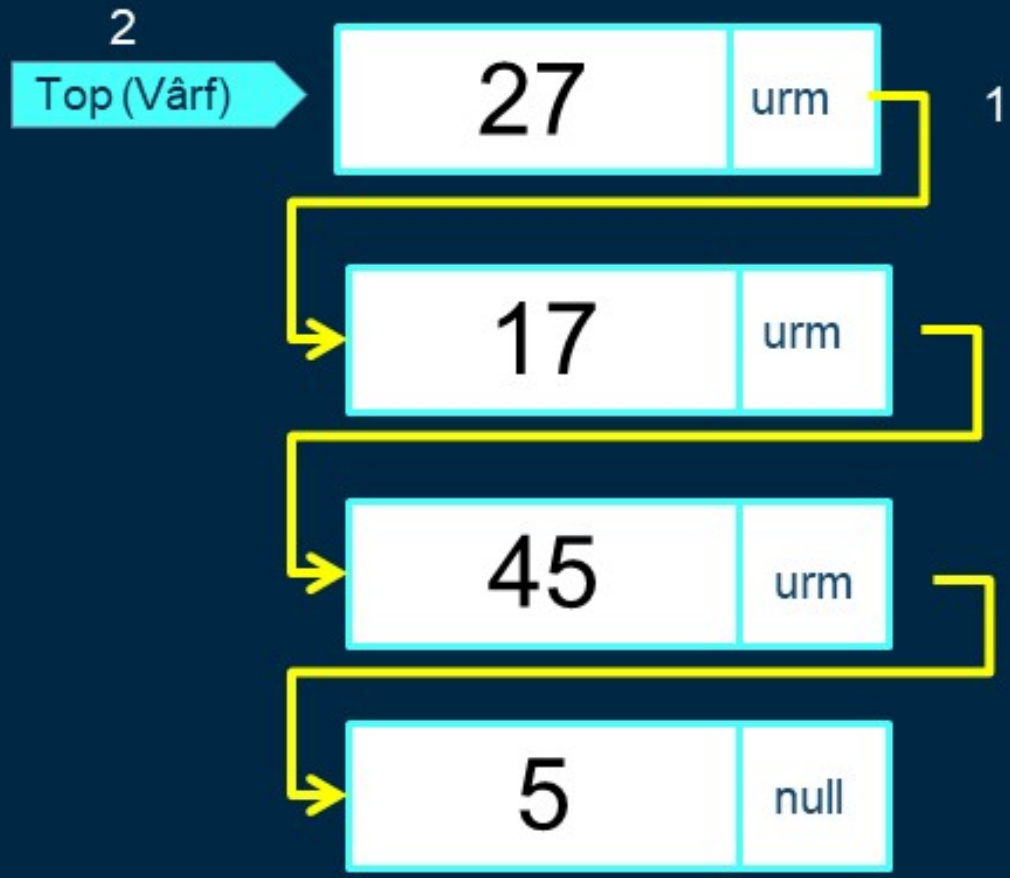
- push() - a introduce un element în stivă



Operatii

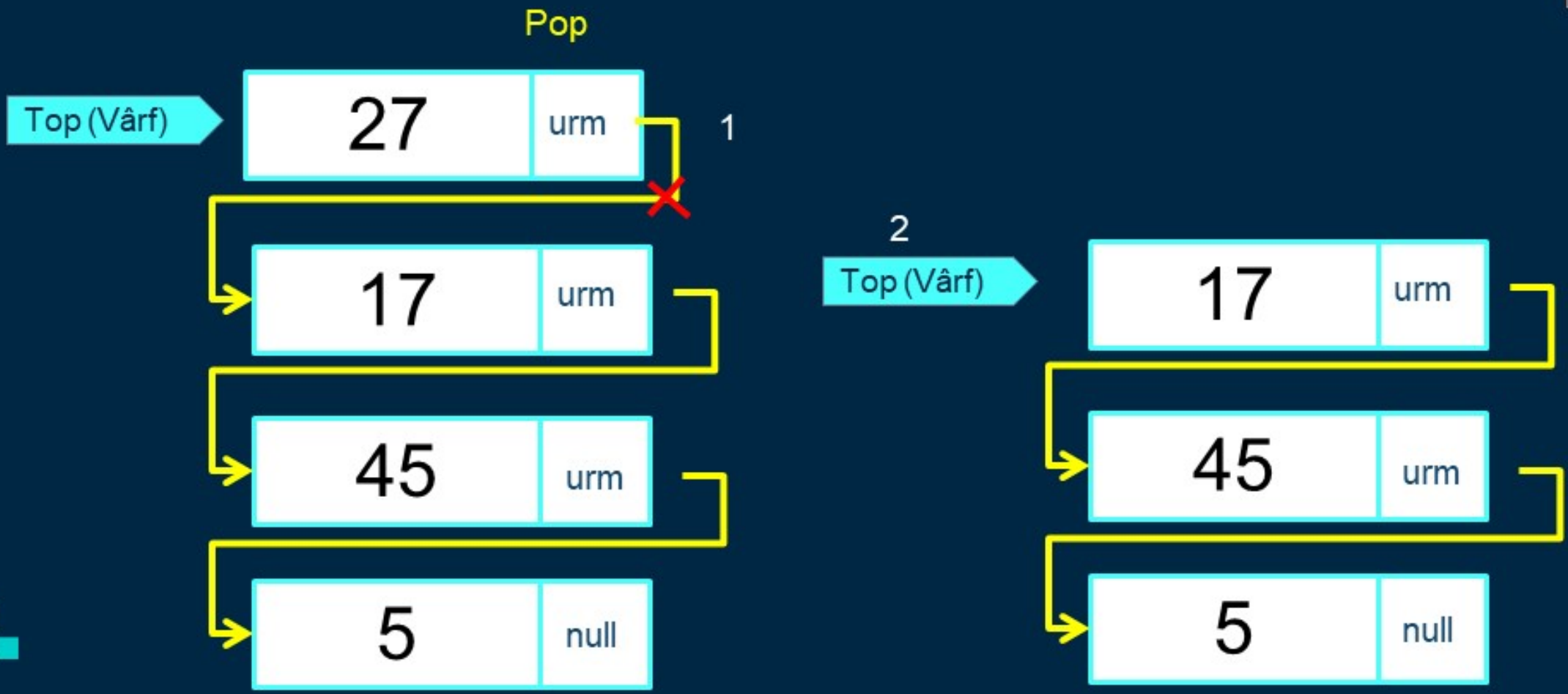
- push() - a introduce un element în stivă

```
begin
if stack is full then
      return
else
      increment top
      stack[top] -> value
      return top
endif
end procedure
```



Operatii

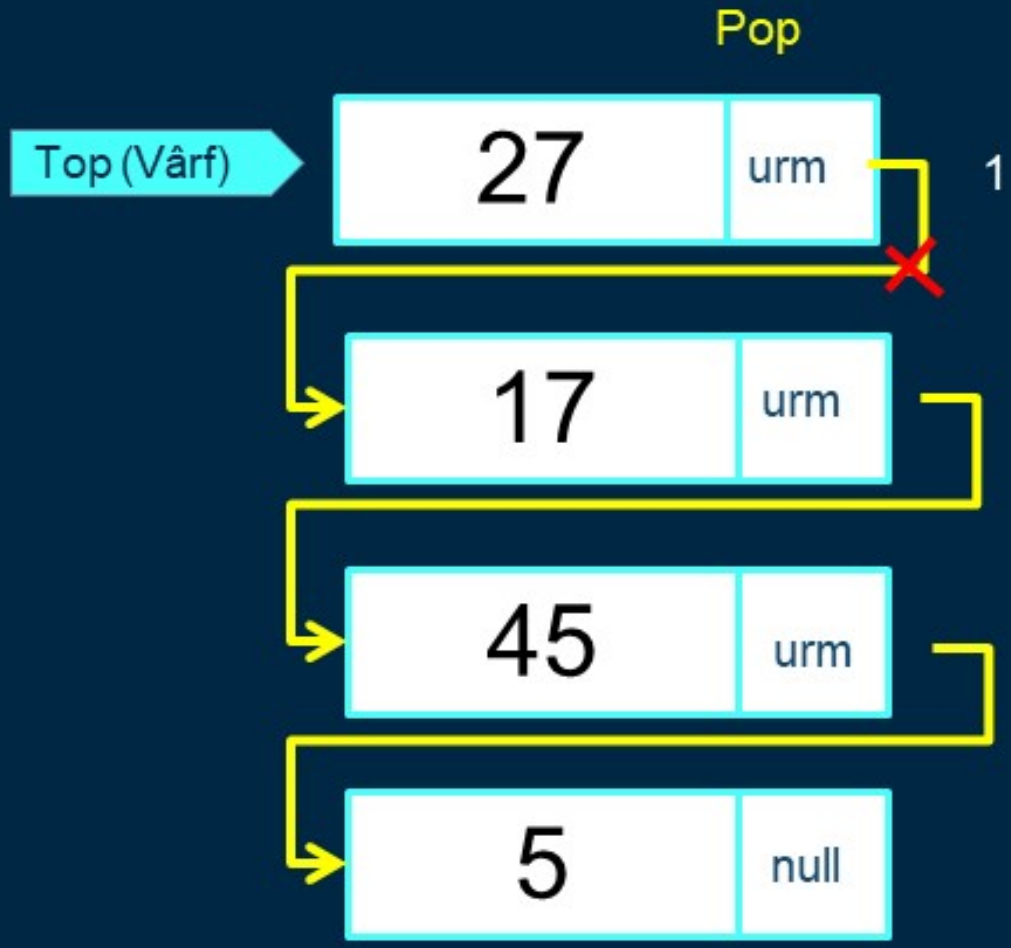
- pop() - a elimina un element din stivă



Operatii

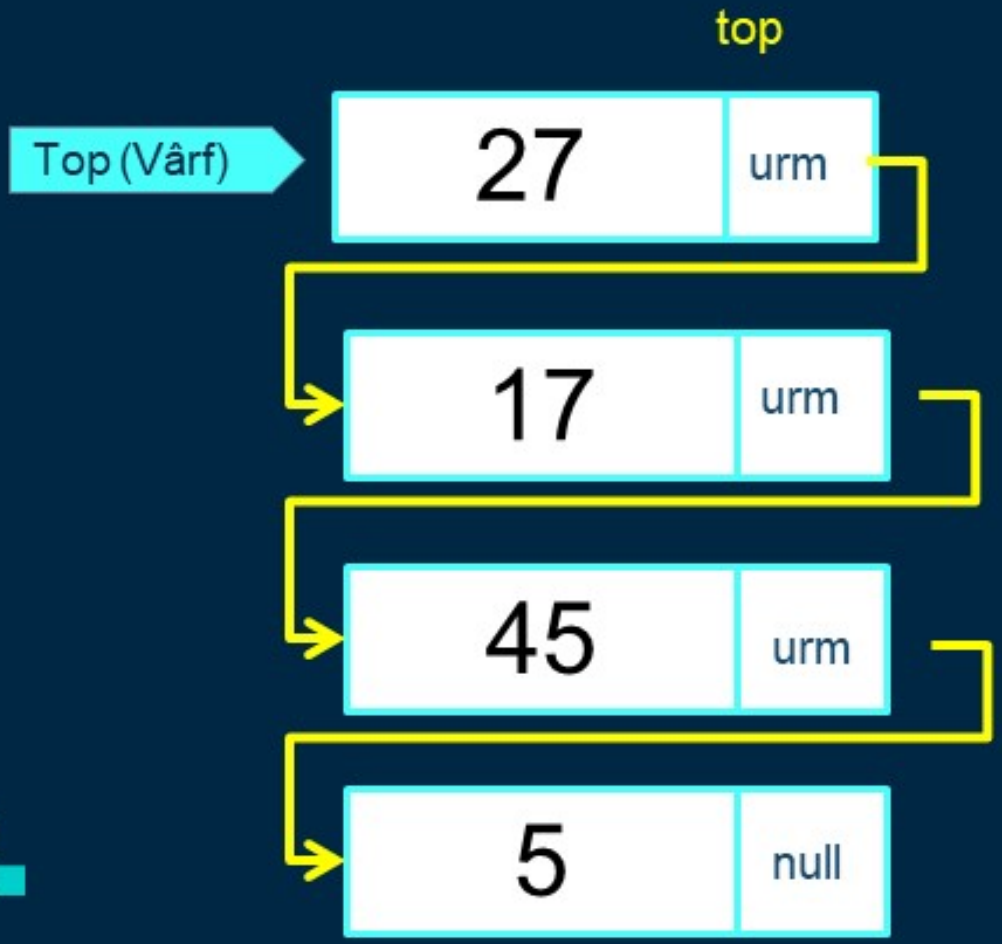
- pop() - a elimina un element din stivă

```
begin
if stack isEmpty then
      return
else
      value -> stack[top]
      decrement top
      return value
endif
end procedure
```



Operatii

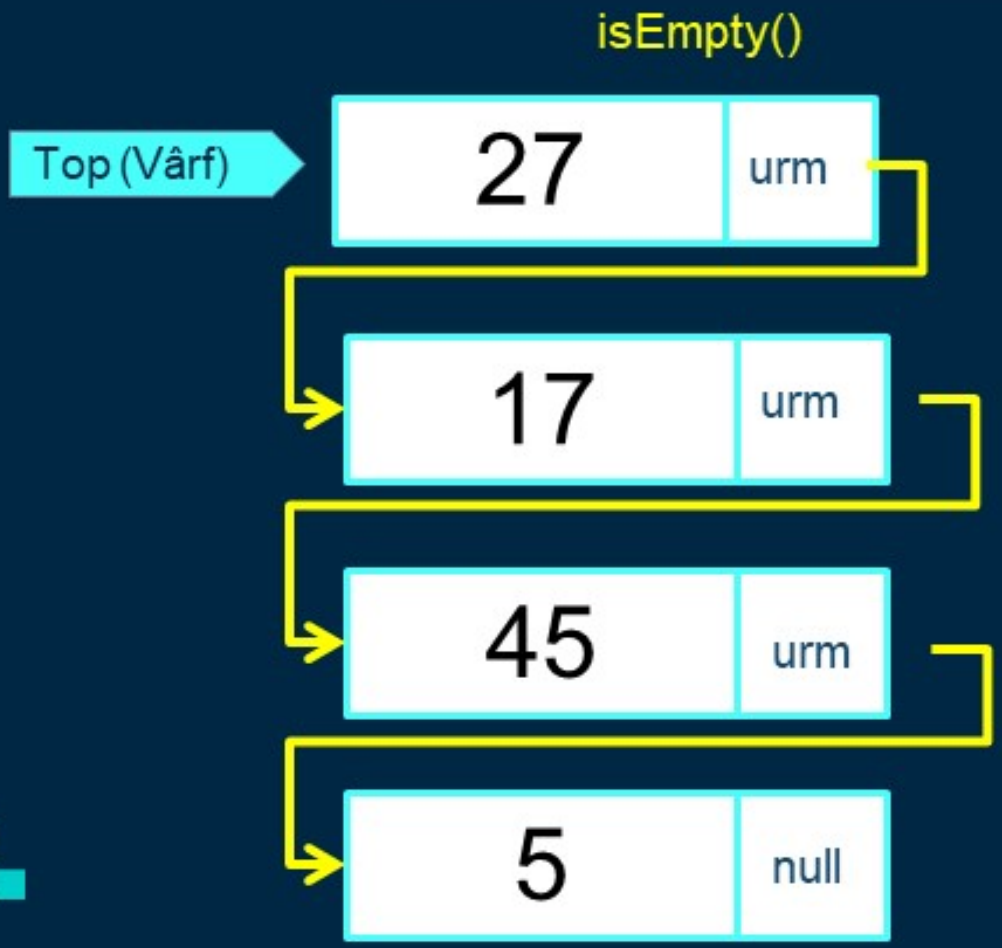
- top() - returneaza elementul din varful stivei




```
begin
   return stack[top]
end procedure
```

Operatii

- isEmpty() - returnează adevărat dacă stiva este goală, altfel false




```
begin
        if top < 1 then
                return true
        else
                return false
        endif
end procedure
```

Complexitate operații Stiva

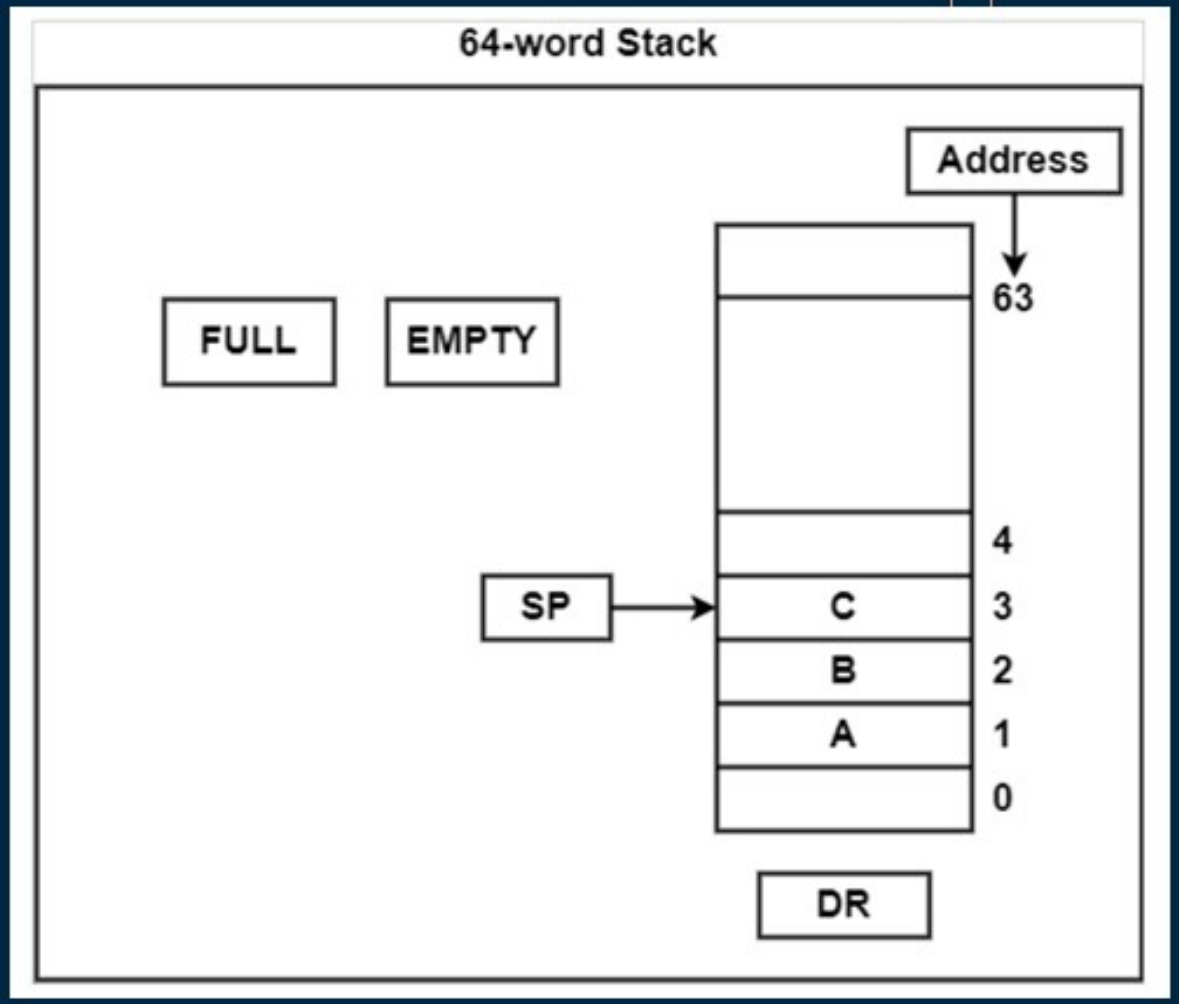
Operația	Complexitate
push()	O(1)
pop()	O(1)
isEmpty()	O(1)
size()	O(1)

Tipuri de stive

- **Stivă registru**: un element de memorie prezent în unitatea de memorie și poate gestiona doar o cantitate melativ redusă de date
- **Stivă de memorie**: poate gestiona o cantitate mare de date de memorie.

Tipuri de stive

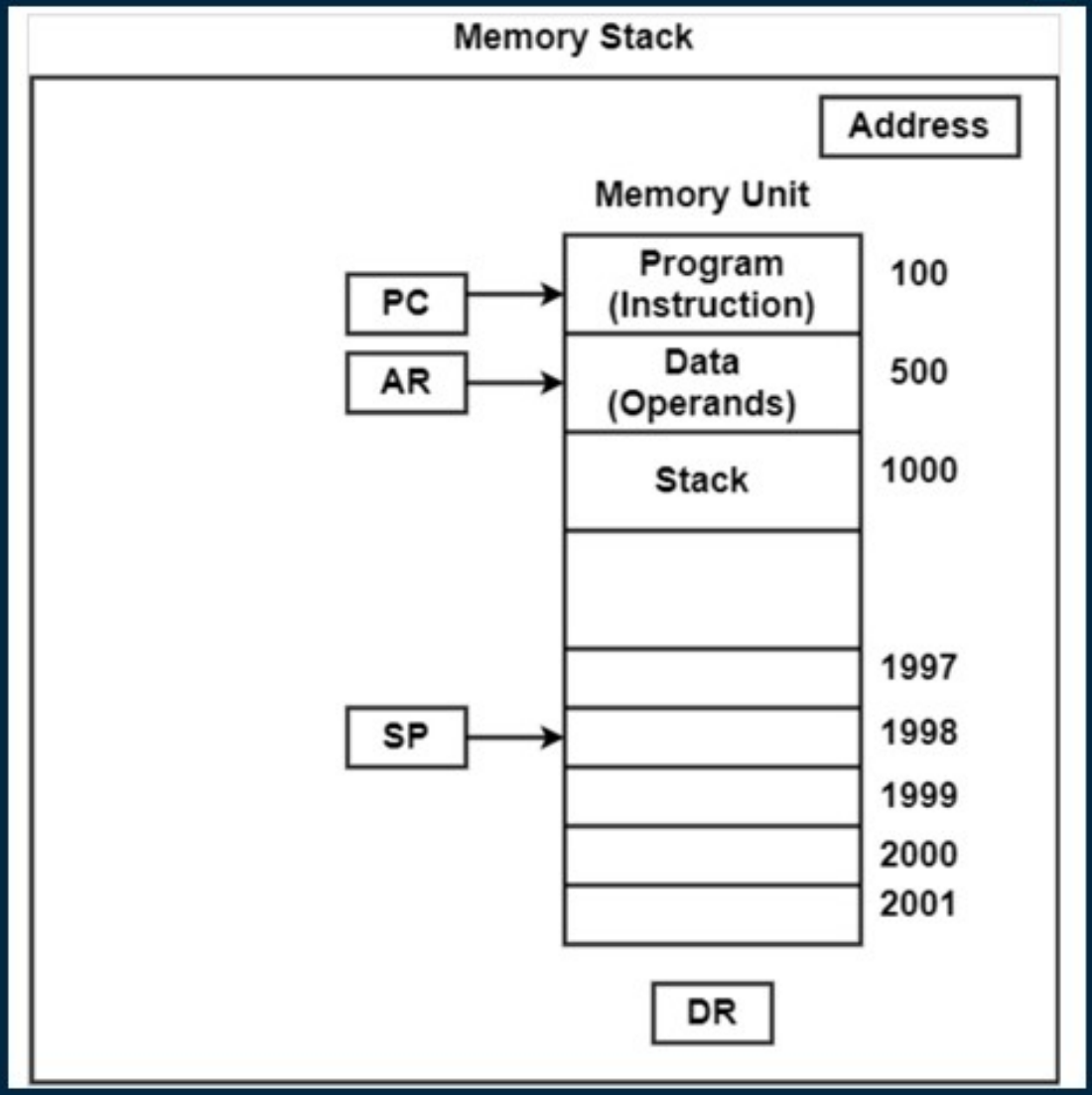
- **Stivă registru**



https://www.tutorialspoint.com/what-is-stack-organization

Tipuri de stive

- **Stivă memorie**



https://www.tutorialspoint.com/what-is-memory-stack-in-computer-architecture

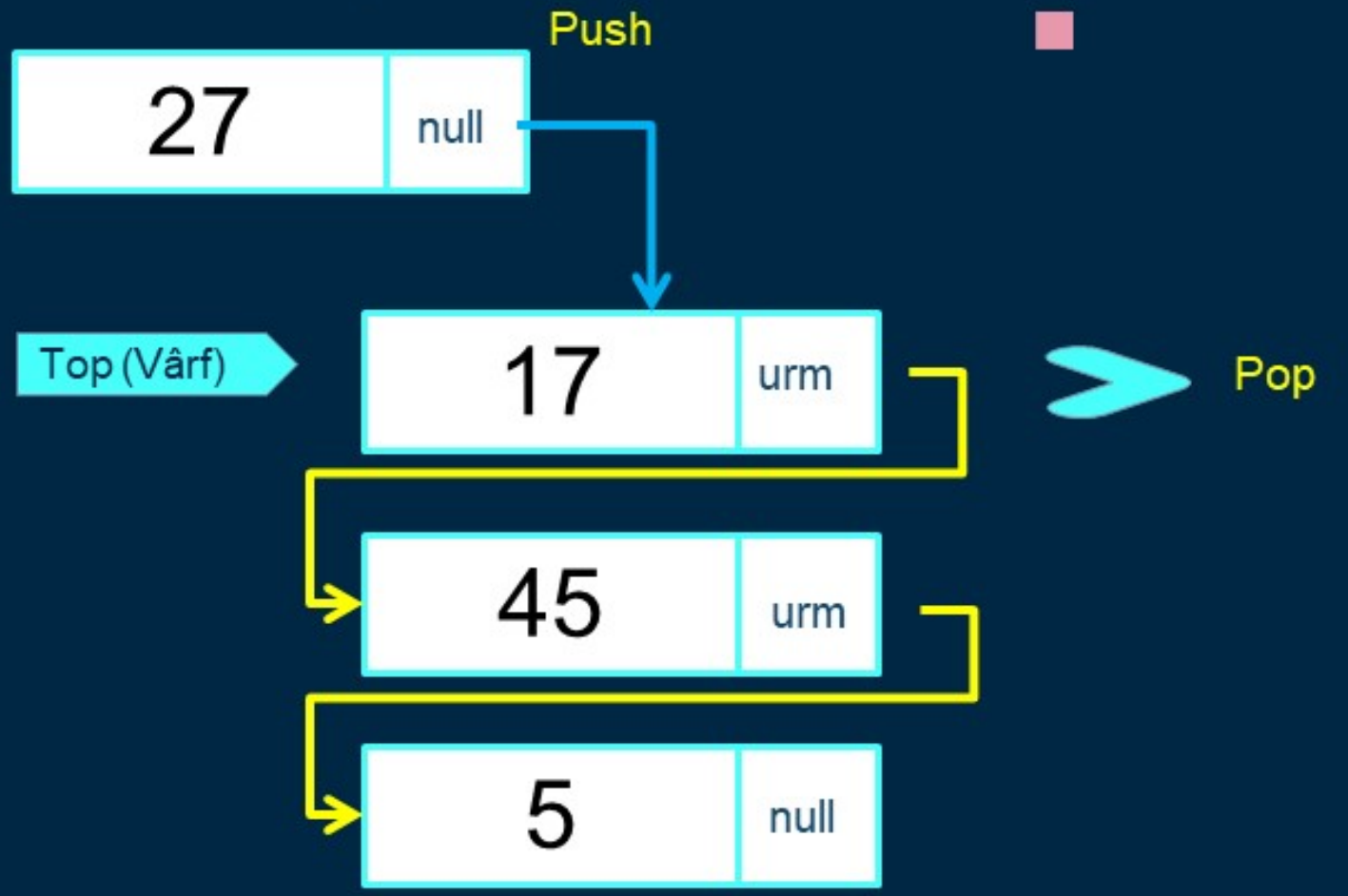
Aplicatii stive

- Conversie Infix în Postfix/Prefix expresii
- Funcții de Undo-Redo, înainte și înapoi în browserele web
- Algoritmi: Turnul din Hanoi, parcurgeri arbori, probleme de stocare și probleme de histogramă, inversare de siruri.
- Backtracking: Knight-Tour, problema N-Queen, găsește-ți drumul printr-un labirint și șah sau dame asemănătoare.
- Sortarea topologică și Componentele puternic conectate (grafuri)
- În gestionarea memoriei

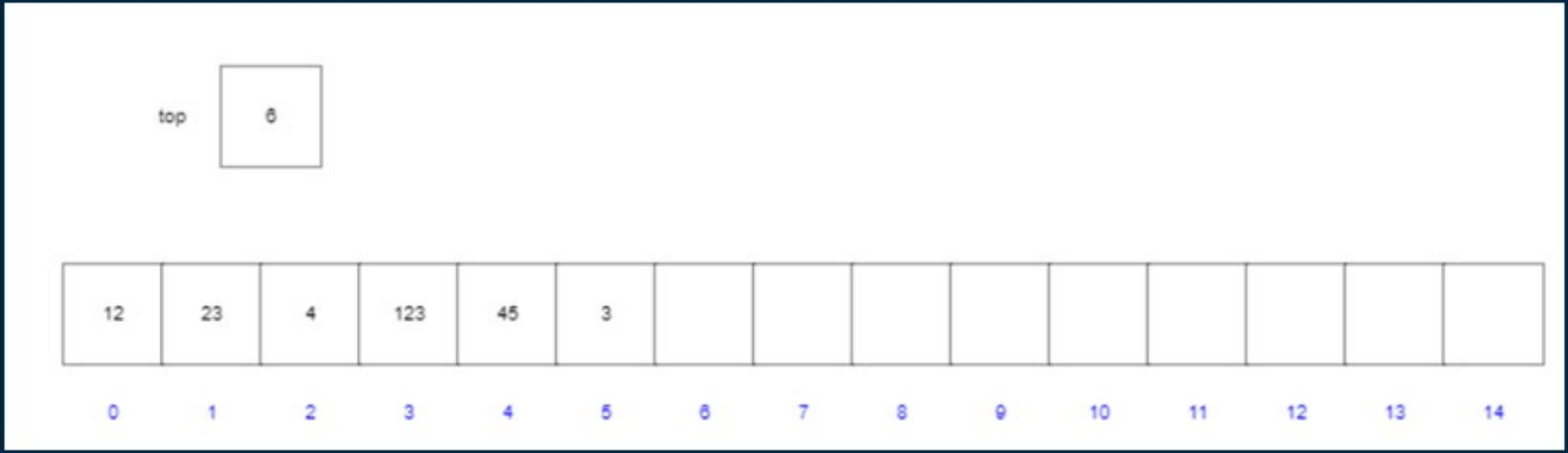
Implementare stive

- Liste înlănțuite
- Tablouri unidimensionali

Stiva – listă înlănțuită



Stiva – tablou unidimensional



top = 6 **12, 23, 4, 123, 45, 3**

Index = top - 1

Coada

02

Coada

Este o structură de date liniară care urmează o anumită ordine în care sunt efectuate operațiunile.

FIFO (Primul intrat, primul ieşit):

elementul care este introdus primul va ieși primul.

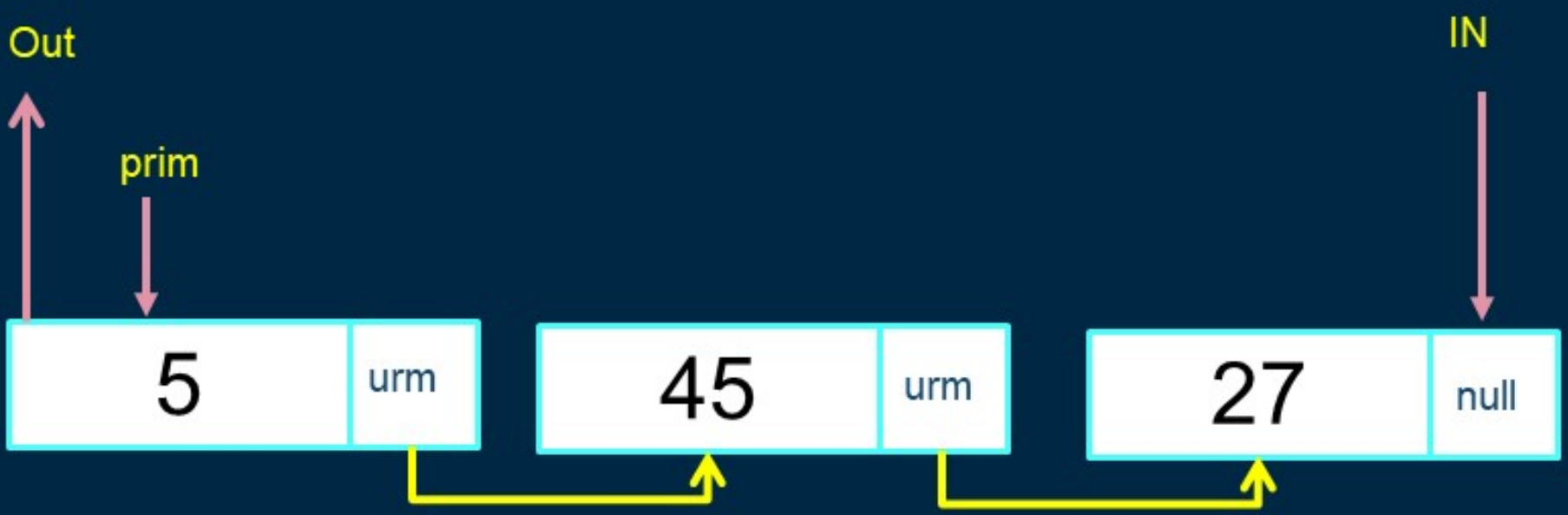
Exemplu:
O coada la casa de bilete. Un sir de masini la intrarea pe un drum cu taxa. Sir de persoana pe scara rulantă.

https://www.tutorialspoint.com/data_structures_algorithms/dsa_queue.htm

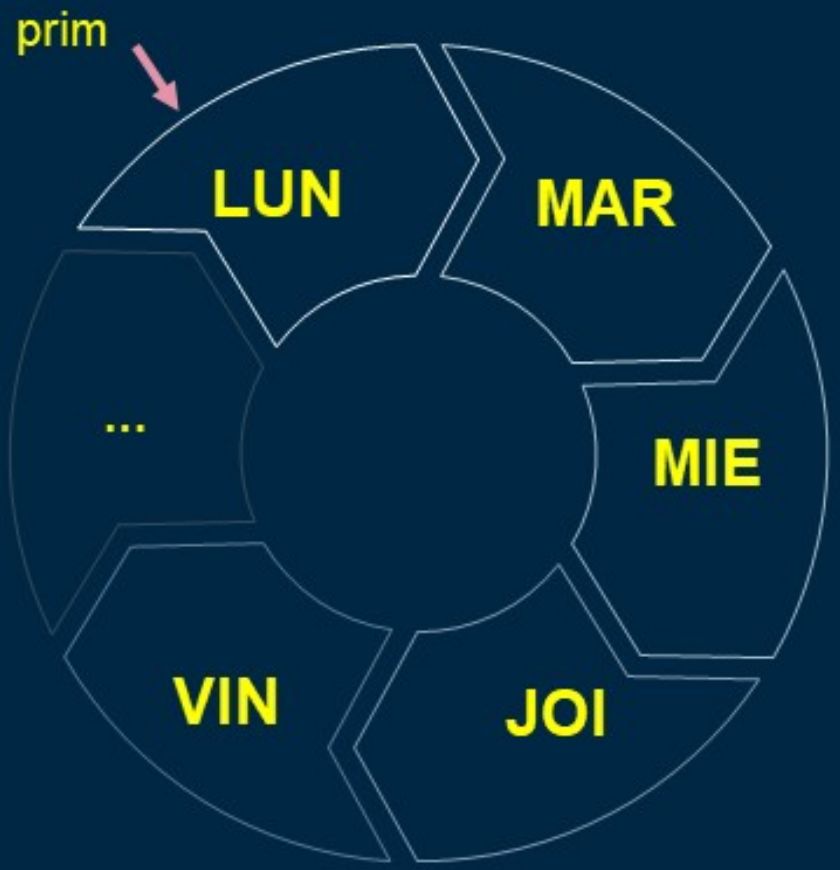
Coada simplă

First In First Out

Last In Last Out



Coada circulară



Ultimul nod este legat de primul nod (Ring Buffer)
Inserarea are loc în partea din față a cozii și ștergerea la sfârșitul acesteia.

Aplicarea cozii circulare: Inserarea zilelor intr-o saptamana.

Coada prioritară

Nodurile listei au o anumită prioritate

Algoritmul lui Dijkstra (cel mai mic drum)
Algoritmul lui Prim
Cod Huffman

https://www.naukri.com/learning/articles/queue-data-structure-types-implementation-applications/

Coada dublu (Dequeue)

Adaugarea si extragerea se poate face la ambele capete

Acțiuni ca stivă și coadă
Stocarea istoricului unui browser web.
Stocarea listei de operațiuni de anulare a unei aplicații software.
Algoritmul de programare a locurilor de muncă

https://www.naukri.com/learning/articles/queue-data-structure-types-implementation-applications/

Operatii

- peek() – redă primul element al cozii fără a-l elimina.
- isfull() – returnează dacă este plină sau nu coada.
- isempty() – verifică dacă este goală sau nu coada.
- add() – adăugare element nou (la final).
- extrag() – extragere prim element din coada.

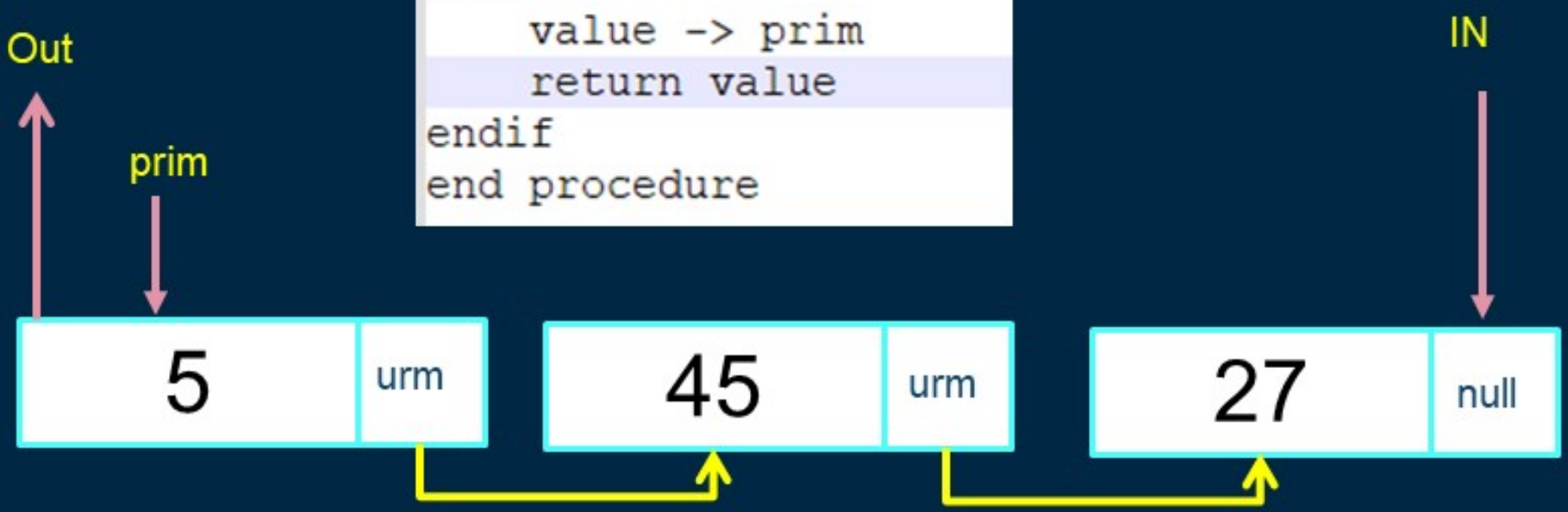
Operatii

- peek() – redă primul element al cozii fără a-l elimina.

First In First Out

```
begin
if queue isEmpty then
    return
else
    value -> prim
    return value
endif
end procedure
```

Last In Last Out



Operatii

- isfull() – returnează dacă este plină sau nu coada.

First In First Out

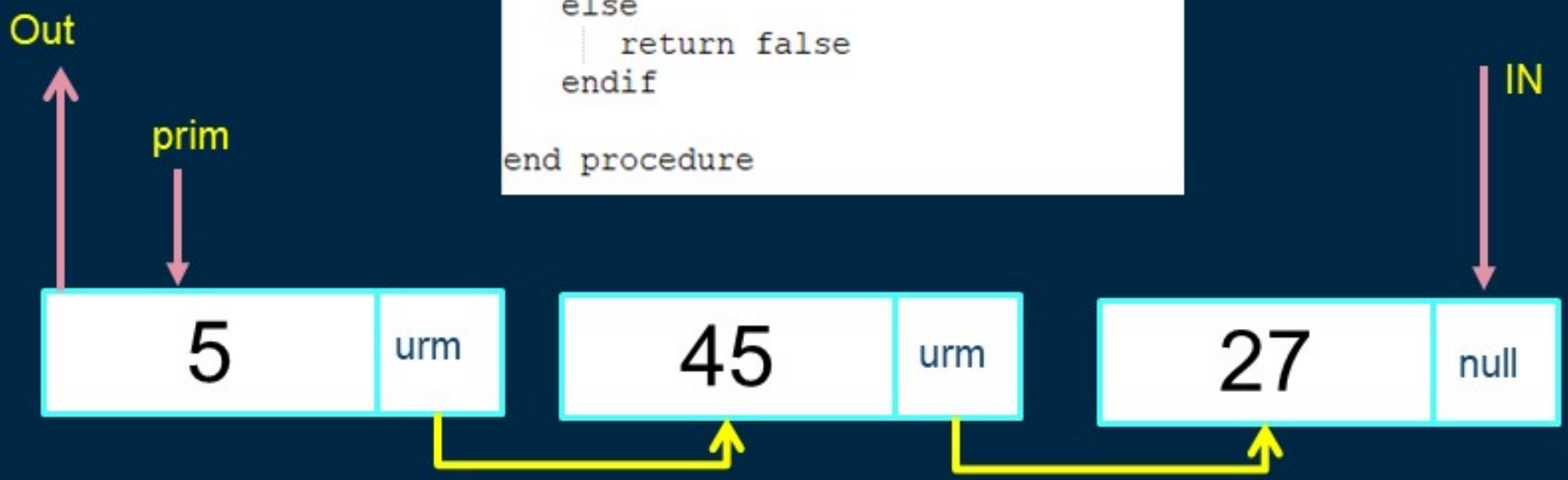
Last In Last Out

```
begin procedure isfull

    if rear equals to MAXSIZE then
        return true
    else
        return false
    endif

end procedure
```

MAXSIZE



Operatii

- isempty() – verifică dacă este goală sau nu coada.

First In First Out

Last In Last Out

```
begin procedure isEmpty

   if front is less than MIN  OR front is greater than rear then
      return true
   else
      return false
   endif

end procedure
```

MAXSIZE

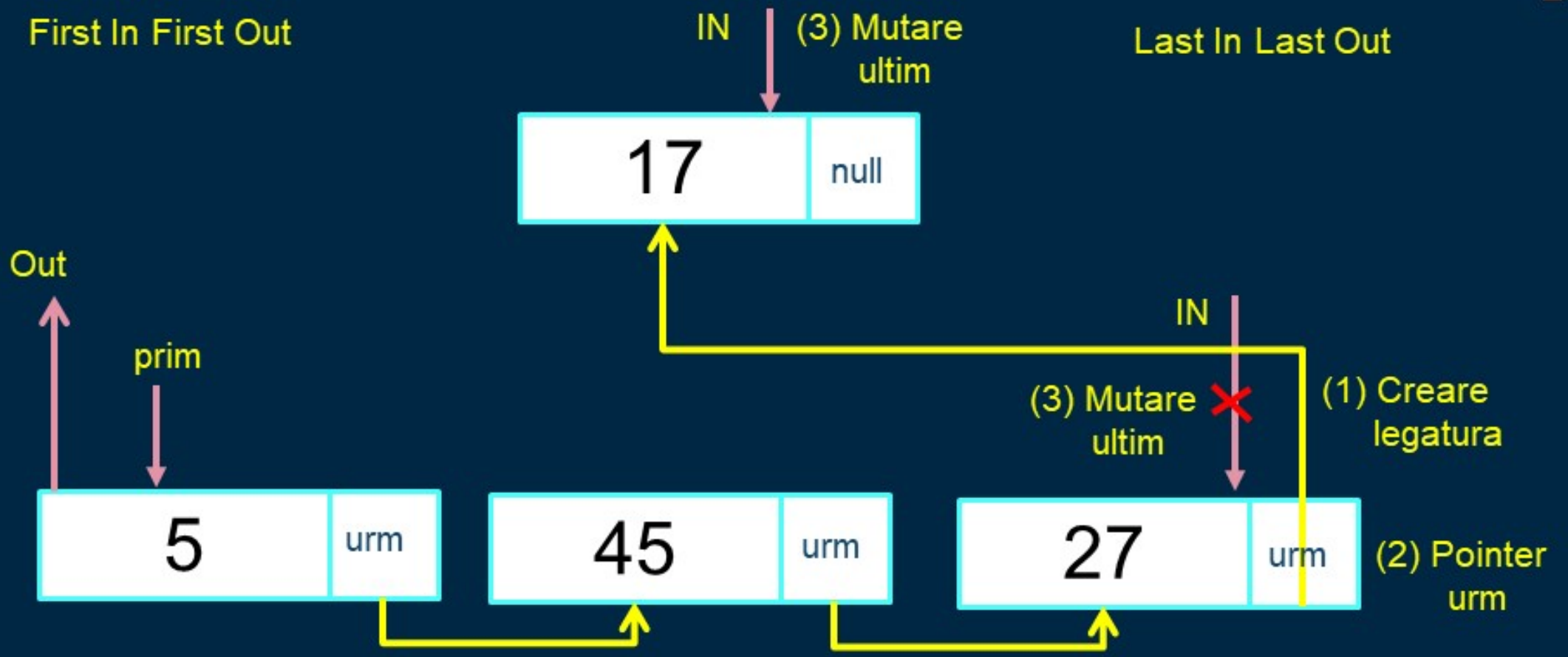
Out

prim

IN

Operatii

- add() – adăugare element nou (la final)

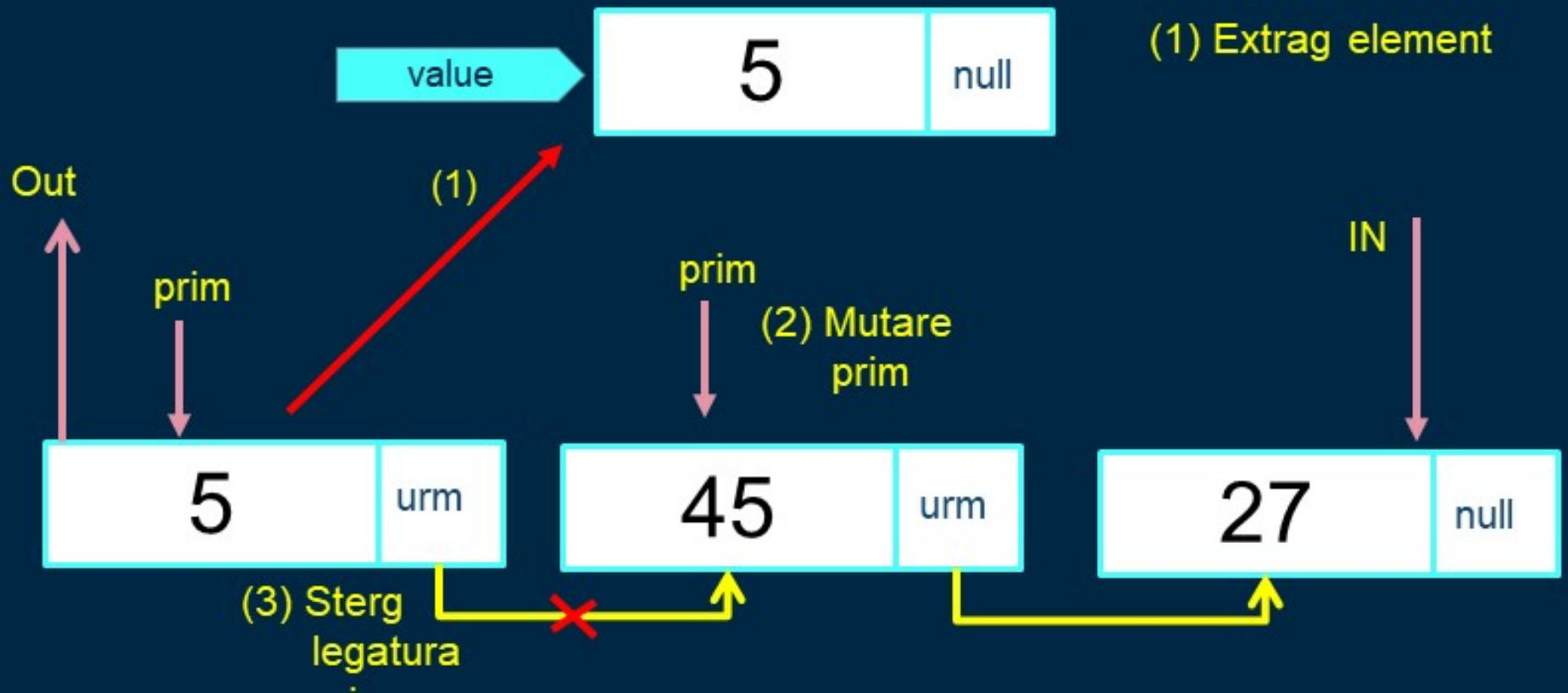


Operatii

- extrag() – extragere prim element din coada.

First In First Out

Last In Last Out



Aplicatii coada

- cozi de sarcini: imprimanta, accesul la stocarea pe disc sau de partajare a timpului, utilizarea procesorului
- o singură resursă și mai mulți consumatori
- sincronizarea între dispozitivele lente și cele rapide
- într-o rețea, o coadă este utilizată în dispozitive precum un router/switch și o coadă de e-mail
- listă de redare piese muzicale (player)

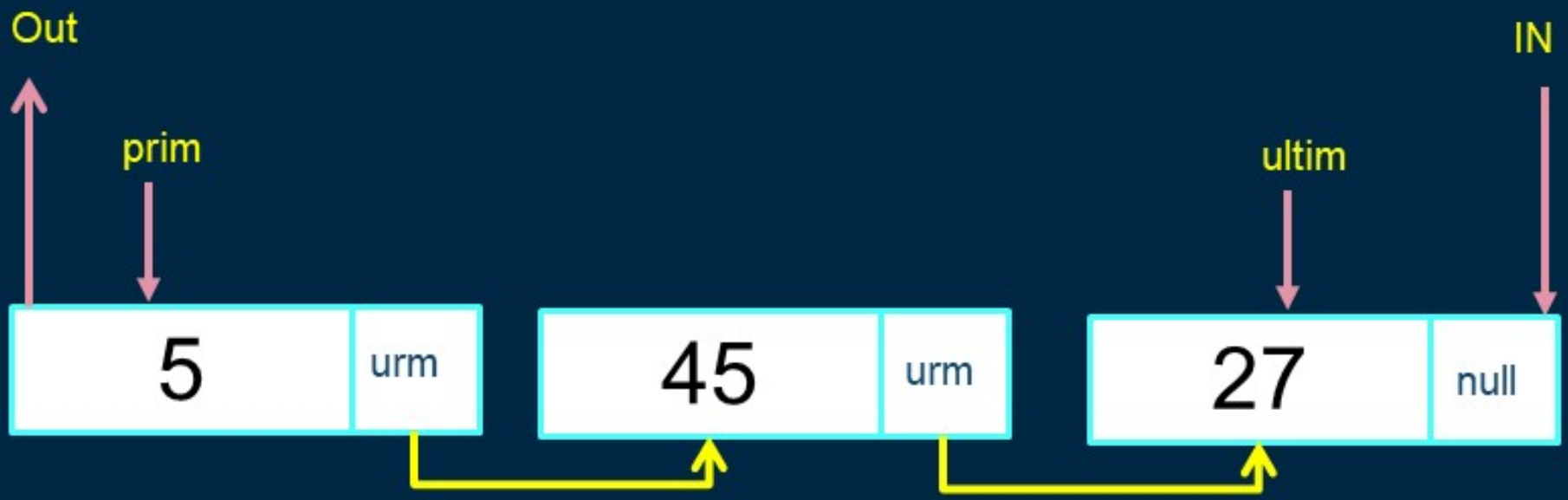
Implementare stive

- Liste înlănțuite
- Tablouri unidimensionali

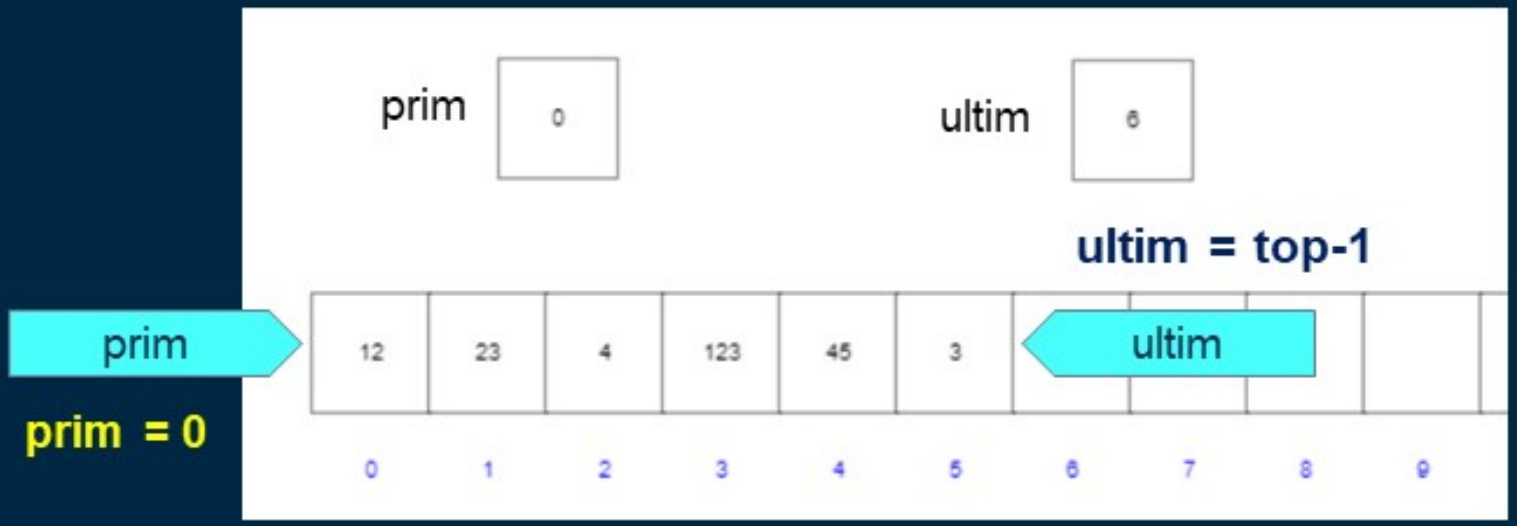
Coada – listă înlănțuită

First In First Out

Last In Last Out



Coada – tablou unidimensional



prim = 0

top = 6

12, 23, 4, 123, 45, 3

Aplicații stiva

IN-fix in POST - fix

9*9/(7-1)+5*6

9 9 * 7 1 - / 5 6 * +

Expresia	Stiva	Ieșire
9	Vidă	9
*	*	9
9	*	9 9
/	/	9 9 *
(	/(	9 9 *
7	/(	9 9 * 7
-	/(-	9 9 * 7
1	/(-	9 9 * 7 1
)	-	9 9 * 7 1 -
+	+	9 9 * 7 1 - /
5	+	9 9 * 7 1 - / 5
*	+*	9 9 * 7 1 - / 5 6
6	+*	9 9 * 7 1 - / 5 6
	Vidă	**9 9 * 7 1 - / 5 6 * +**

Aplicații stiva

INVERSARE

Fereastră

ărtsaereF

Expresia	Stiva
	Vidă
F	F
e	e F
r	
e	
a	
s	
t	
r	
ă	ă r t s a e r e F
	Vidă

Aplicații stiva

Implementare C

```c
#include<stdio.h>
#define size 5
int arr[size];
int top = -1;          //-1 stiva vida
int estePlinaStiva()
{
    if(top == size - 1)
        return 1;
    return 0;
}

void push(int val)
{
   if(estePlinaStiva())
        printf("Nu mai pot adauga elementul %d. Stiva este plina!\n",val);
    else
    {
        ++top;
        arr[top]=val;
    }
}
```

Aplicații stiva

Implementare C

```c
int esteVidaStiva()
{
    if(top == -1)
        return 1;
    return 0;
}

void pop()
{
    if(esteVidaStiva())
        printf("Stiva este vida\n");
    else
    {
        printf("Element adaugat = %d\n",arr[top]);
        top--;
    }
}
```

Aplicații stiva

Implementare C

```c
int main()
{
    push(100);
    push(130);
    push(150);
    push(200);
    push(1100);
    push(125);
    pop();
    pop();
    pop();
    pop();
    pop();
    pop();
    return 0;
}
```

```c
#include<stdio.h>
#define size 5
int arr[size];
int top = -1;          //-1 stiva vida
```

```
Nu mai pot adauga elementul 125. Stiva este plina!
Element adaugat = 1100
Element adaugat = 200
Element adaugat = 150
Element adaugat = 130
Element adaugat = 100
Stiva este vida
```

Intrebari?

dorin.lordache@955.univ-ovidius.ro

Mulțumesc