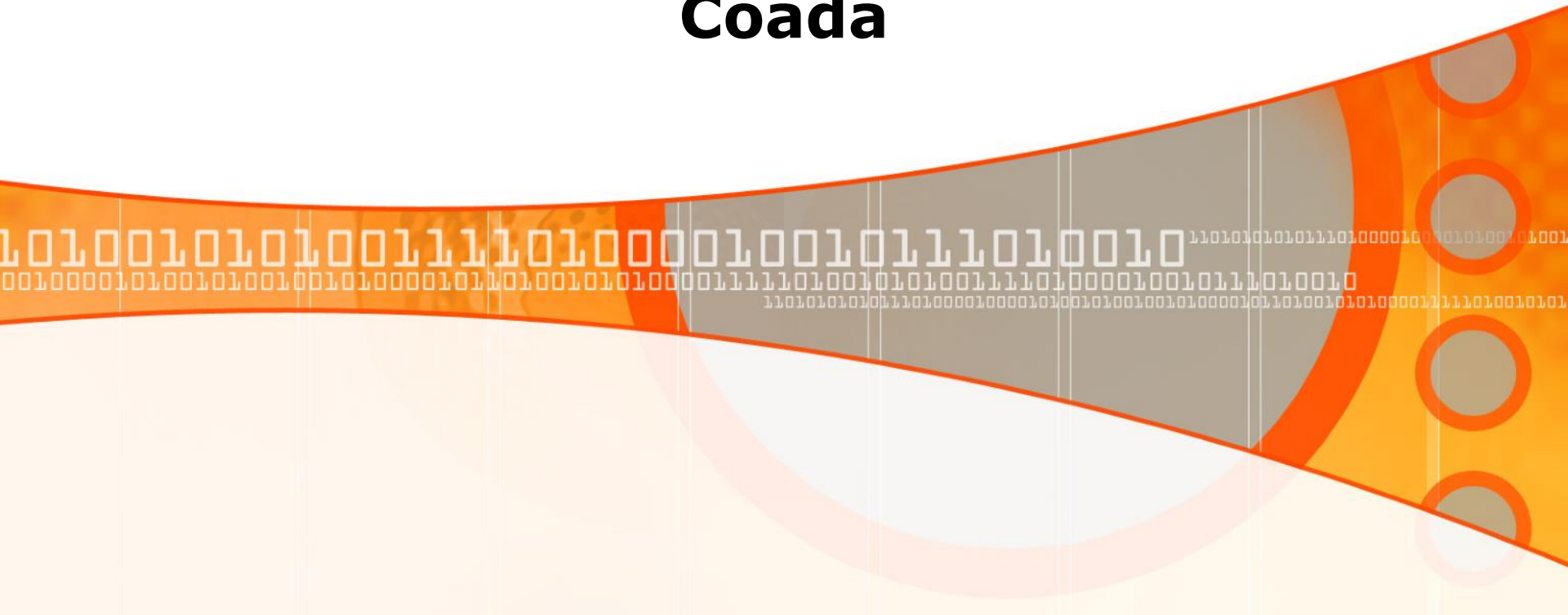


# STRUCTURI DE DATE

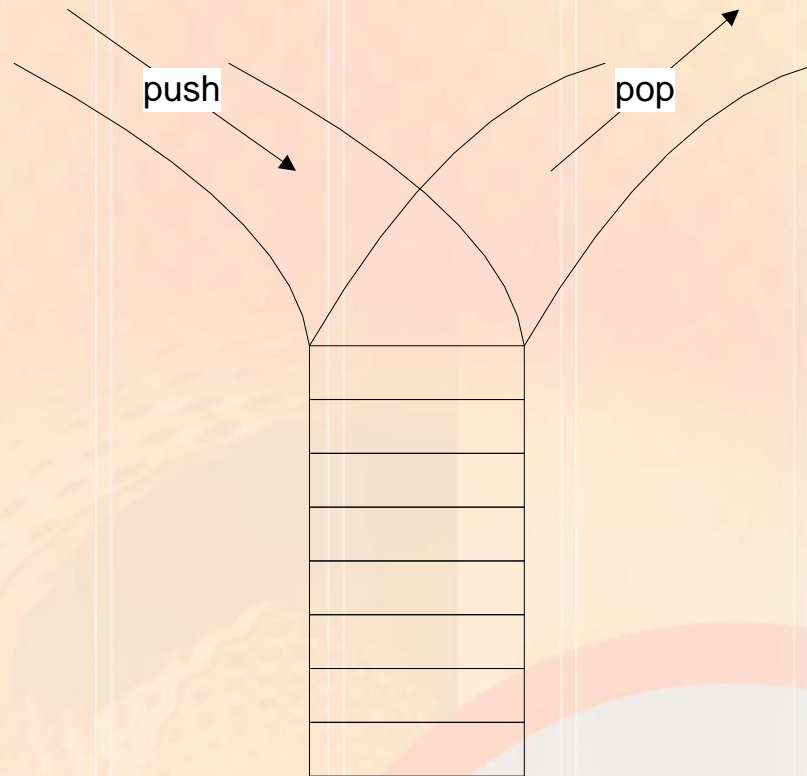
**Stiva**  
**Coadă**



# STIVA

## Structura de tip stiva:

- Structura de date **logica**: implementarea este făcută utilizând alte structuri de date;
- Structura de date **omogena**: 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 (inceput lista simpla), denumit **vârful stivei**.

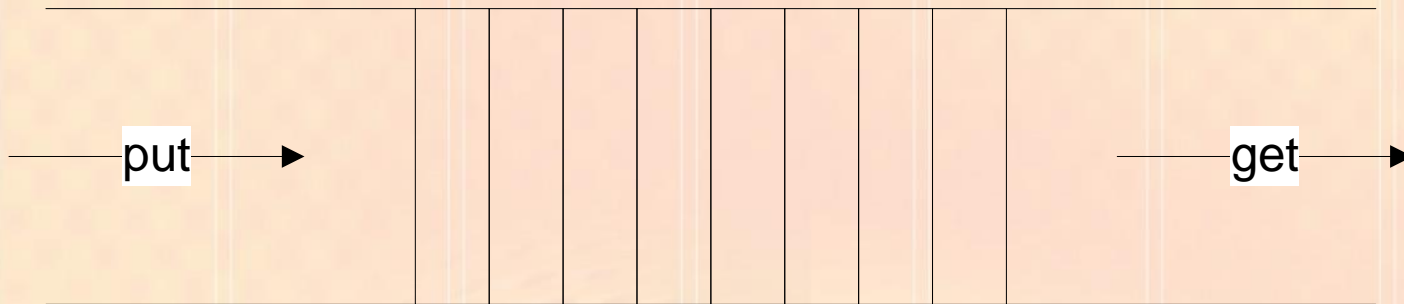


***Mecanismul de stiva***

# COADA

## Structura de tip coada:

- Structura de date **logica**: implementarea este făcută utilizând alte structuri de date;
- Structura de date **omogena**: 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 capat (sfarsit lista simpla) și extragerile (get) sunt făcute la celalalt capat (inceput lista simpla).



## Mecanismul de coadă

## Conversia unei valori zecimale in format binar

1. Preluare valoare zecimala;
2. Cat timp valoarea este strict pozitiva:
  - 2.1 determinare rest impartire la 2;
  - 2.2 prelucrare rest – utilizare structura stiva;
  - 2.3 determinare cat impartire la 2.



# STIVE SI COZI - APLICATII

## QuickSort

- Algoritm de tip *divide-et-impera*;
- Reducerea fiecarui set de valori de sortat in 2 sub-seturi;
- Aplicarea operatiei de reducere pe fiecare din cele 2 sub-seturi;
- Utilizarea a 2 structuri de tip stiva (*Inf*, *Sup*) pentru stocarea pozitiilor celor 2 sub-seturi.
- Stivele sunt initializate: *Inf* = {1}, *Sup* = {10}.

# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

Faza de reducere:

- extragere varf de stiva *Inf* (index **1**) si varf de stiva *Sup* (index **10**)
- identificare pozitie finala a unei valori din set (ex. valoare pozitie **1**).

<b>13</b>	85	25	9	64	99	93	49	72	20
-----------	----	----	---	----	----	----	----	----	----

Incepand cu pozitia **10**, se cauta de la dreapta la stanga prima valoare mai mica decat **13**.



# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

Valoare identificata este **9**.

<b>13</b>	85	25	<b>9</b>	64	99	93	49	72	20
-----------	----	----	----------	----	----	----	----	----	----

Se interschimba **13** cu **9**.

<b>9</b>	85	25	<b>13</b>	64	99	93	49	72	20
----------	----	----	-----------	----	----	----	----	----	----

Incepand cu pozitia lui **9**, se cauta de la stanga spre dreapta prima valoare mai mare decat **13**,  
pana la pozitia lui **13**.

# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

Valoare identificata este **85**.

9	<b>85</b>	25	<b>13</b>	64	99	93	49	72	20
---	-----------	----	-----------	----	----	----	----	----	----

Se interschimba **85** cu **13**.

9	<b>13</b>	25	<b>85</b>	64	99	93	49	72	20
---	-----------	----	-----------	----	----	----	----	----	----

Incepand cu pozitia lui **85**, se cauta de la dreapta la stanga prima valoare mai mica decat **13**,  
**pana la pozitia lui 13**.

# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

Nu exista nici o valoare, deci pozitia finala a lui **13** este determinata (**2**).

9	<b>13</b>	25	85	64	99	93	49	72	20
---	-----------	----	----	----	----	----	----	----	----

Stivele sunt populate pentru cele 2 subintervale.

*Subintervalul stang* contine 1 element, deci este sortat. Nu se retine nimic pe stive.

*Subintervalul drept* contine cel putin 2 elemente.

Stivele sunt: ***Inf*** = {**3**} si ***Sup*** = {**10**}.

# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

Faza de reducere:

- extragere varf de stiva *Inf* (index **3**) si varf de stiva *Sup* (index **10**)
- identificare pozitie finala a primei valori din set **25**.

9	13	25	85	64	99	93	49	72	20
---	----	----	----	----	----	----	----	----	----

Incepand cu pozitia **10**, se cauta de la dreapta la stanga prima valoare mai mica decat **25**.

# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

Valoare identificata: **20**.

9	13	25	85	64	99	93	49	72	20
---	----	----	----	----	----	----	----	----	----

Se interschimba **25** cu **20**.

9	13	20	85	64	99	93	49	72	25
---	----	----	----	----	----	----	----	----	----

Incepand cu pozitia lui **20**, se cauta de la stanga la dreapta prima valoare mai mare decat **25**.



# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

Valoare identificata: **85**.

9	13	20	<b>85</b>	64	99	93	49	72	<b>25</b>
---	----	----	-----------	----	----	----	----	----	-----------

Se interschimba **25** cu **85**.

9	13	20	<b>25</b>	64	99	93	49	72	<b>85</b>
---	----	----	-----------	----	----	----	----	----	-----------

Incepand cu pozitia lui **85**, se cauta de la dreapta la stanga prima valoare mai mica decat **25**.

# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

Valoare identificata: -.

9	13	20	25	64	99	93	49	72	85
---	----	----	----	----	----	----	----	----	----

Este determinata pozitia **25** in setul final (ordonat),  
respectiv pozitia **4**.

Stivele sunt populate pentru cele 2 subintervale.

*Subintervalul stang* contine 1 element, deci este sortat.  
Nu se retine nimic pe stive.

*Subintervalul drept* contine cel putin 2 elemente. Stivele  
sunt: ***Inf* = {5}** si ***Sup* = {10}**.

# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

Faza de reducere:

- extragere varf de stiva *Inf* (index **5**) si varf de stiva *Sup* (index **10**)
- identificare pozitie finala a primei valori din set **64**.

9	13	20	25	64	99	93	49	72	85
---	----	----	----	----	----	----	----	----	----

Incepand cu pozitia **10**, se cauta de la dreapta la stanga prima valoare mai mica decat **64**.

# STIVE SI COZI - APLICATII

**QuickSort** – exemplu:

- Faza de reducere se aplica recursiv;
- Opreire algoritm: stivele *Inf* si *Sup* sunt goale;
- Stivele sunt utilizate pentru a retine limitele de intervale obtinute din modul recursiv de aplicare a algoritmului.

# STIVE SI COZI - APLICATII

**Evaluarea expresiilor matematice** ce utilizeaza 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;



# STIVE SI COZI - APLICATII

Evaluarea expresiilor matematice  
(continuare):

- Forma poloneză: scrierea operatorilor înaintea operanzilor;
- Forma poloneză inversă: operatorii sunt scriși în urma operanzilor.

# STIVE SI COZI - APLICATII

Forma poloneză inversă (scriere postfixata):  
avantaje față de scrierea prefixată (forma poloneza) sau infixată (expresia matematica):

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

# STIVE SI COZI - APLICATII

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 după cum este prezentat în tabelul.

# STIVE SI COZI - APLICATII

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

Ierarhia operatorilor

# STIVE SI COZI - APLICATII

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

Forme ale scrierii unei expresii matematice



# STIVE SI COZI - APLICATII

Algoritmul 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 „(”;

# STIVE SI COZI - APLICATII

- altfel:
  - a. atâta timp cât ierarhia operatorului din vârful stivei este mai mare ierarhia operatorului curent, se trece elementul din vârful stivei în scrierea postfixată;
  - b. se introduce operatorul curent în stivă.
- se trec toți operatorii rămași pe stivă în scrierea postfixată.

# STIVE SI COZI - APLICATII

Algoritmul de evaluare:

- 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):
    - a. se extrage din stivă elementul y;
    - b. se extrage din stivă elementul x;
    - c. se efectuează operația x operator y;
    - d. se depune rezultatul pe stivă;
- ultima valoare care se află pe stivă este rezultatul expresiei.

# STIVE SI COZI - APLICATII

La nivel de sistem de operare:

- Coada procese care asteapta producerea unui eveniment;
- Implementare buffer de mesaje intre procese/sisteme sub forma de coada;
- Zona de memorie utilizata pentru stocarea datelor din variabile locale, argumente ale functiilor, adrese de revenire in apelul superior de functie, rezultate stocate temporar etc;

# STIVE SI COZI - APLICATII

**Suport pentru implementarea/utilizarea altor structuri de date (Abstract Data Type) prin stabilirea ordinii fazelor de prelucrare (ex. grafuri).**