TAD Coada (QUEUE)

Observaţii:

1. În limbajul uzual cuvântul "coadă" se referă la o înşiruire de oameni, maşini, etc., aranjaţi în ordinea sosirii şi care aşteaptă un eveniment sau serviciu.
   - Noii sosiţi se poziţionează la sfârşitul cozii.
   - Pe măsură ce se face servirea, oamenii se mută câte o poziţie înainte, până când ajung în faţă şi sunt serviţi, asigurându-se astfel respectarea principiului "primul venit, primul servit".
   - Exemple de cozi sunt multiple: coada de la benzinării, coada pentru cumpărarea unui produs, etc. Tipul de date **Coadă** permite implementarea în aplicaţii a acestor situaţii din lumea reală.
2. O *coadă* este o structură liniară de tip listă care restricţionează adăugările la un capăt şi ştergerile la celălalt capăt (lista FIFO - *First In First Out*).
3. **Accesul** într-o coadă este *prespecificat* (se poate accesa doar elementul cel mai devreme introdus în coadă), nu se permite accesul la elemente pe baza poziţiei. Dintr-o coadă se poate **şterge** elementul CEL MAI DEVREME introdus (primul).
4. Se poate considera şi o capacitate iniţială a cozii (număr maxim de elemente pe care le poate include), caz în care dacă numărul efectiv de elemente atinge capacitatea maximă, spunem că avem o *coadă plină*.
   - adăugarea în coada plină se numeşte **depăşire superioară**.
5. O coadă fără elemente o vom numi *coadă vidă* şi o notăm Φ.
   - ştergerea din coada vidă se numeşte **depăşire inferioară**.
6. O coadă în general nu se iterează.
7. Cozile sunt frecvent utilizate în programare - crearea unei cozi de aşteptare a task-urilor într-un sistem de operare.
   - dacă task-urile nu au asociată o prioritate, ele sunt procesate în ordinea în care intră în sistem $\rightarrow$ **Coadă**.
   - dacă task-urile au asociate o prioritate şi trebuie procesate în ordinea priorităţii lor $\rightarrow$ **Coadă cu priorităţi**.

Tipul Abstract de Date COADA:

domeniu: $\mathcal{C} = \{c \mid c \text{ este o coadă cu elemente de tip } TElement\}$

operaţii:

- creeaza(c)
  {creează o coadă vidă}
  $pre:$ $true$
  $post:$ $c \in \mathcal{C}, c = \Phi$(coada vidă)

- adauga(c, e)
  {se adaugă un element la sfârşitul cozii}
  $pre:$ $c \in \mathcal{C}, e \in TElement, c$ nu e plină
  $post:$ $c' \in \mathcal{C}, c' = c \oplus e,$ e va fi cel mai recent element introdus în coadă

  @ aruncă excepţie dacă coada e plină

- sterge(c, e)
  {se şterge primul element introdus în coadă}
  $pre:$ $c \in \mathcal{C}, c \neq \Phi$
  $post:$ $e \in TElement,$ e este elementul cel mai devreme introdus în coadă, $c' \in \mathcal{C}, c' = c \ominus e$

  @ aruncă excepţie dacă coada e vidă

- element(c, e)
  {se accesează primul element introdus în coadă}
  $pre:$ $c \in \mathcal{C}, c \neq \Phi$
  $post:$ $c' = c, e \in TElement,$ e este elementul cel mai devreme introdus în coadă

  @ aruncă excepţie dacă coada e vidă

- vida (c)
  $pre:$ $c \in \mathcal{C}$
  $post:$ $vida = \begin{cases} adev, & \text{dacă } c = \Phi \\ fals, & \text{dacă } c \neq \Phi \end{cases}$

- plina (c)
  $pre:$ $c \in \mathcal{C}$
  $post:$ $plina = \begin{cases} adev, & \text{dacă } c \text{ e plină} \\ fals, & \text{contrar} \end{cases}$

- distruge(c)
  {destructor}
  $pre:$ $c \in \mathcal{C}$
  $post:$ c a fost 'distrusa' (spaţiul de memorie alocat a fost eliberat)

Observaţii

- Coada nu este potrivită pentru aplicaţiile care necesită traversarea ei (nu avem acces direct la elementele din interiorul cozii).
- Afişarea conţinutului cozii poate fi realizată folosind o coadă auxiliară (scoatem valorile din coadă punându-le într-o coadă auxiliară, după care se reintroduc în coada iniţială).

Implementări ale cozilor folosind

- tablouri - vectori (dinamici) - reprezentare circulară (Figura 1).
- liste înlănţuite.

1	2	3	4	5	6	7	8	9	10	11	12
						15	6	9	8	4	

Față **Spate**

n = 12 (capacitatea tabloului)
Față = 7 - indicele unde e memorat primul element din coadă
Spate = 12 – primul indice liber din spatele cozii

- Se adaugă în **Spate**, se șterge din **Față**
- Elementele cozii se află între indicii **Față, Față+1,...Spate-1**
- Initializarea cozii: **Față=Spate=1**
- Depășire inferioară (coada vidă): **Față=Spate**
- Depășire superioară (coada plină): a) **Față= 1** și **Spate= n** sau b) **Față=Spate+1**

a) **Față= 1** și **Spate= 12**

1	2	3	4	5	6	7	8	9	10	11	12
2	8	9	3	5	7	15	6	9	8	4	

a) **Față= 7** și **Spate= 6**

1	2	3	4	5	6	7	8	9	10	11	12
2	8	9	3	5		15	6	9	8	4	5

Generalizare a cozilor

- **Coada completă** (*Double ended queue* - **DEQUEUE**) - adăugări, ştergeri se pot face la ambele capete ale cozii.

Aplicaţie

Translatarea unei expresii aritmetice din forma infixată în forma postfixată. Fie E o expresie aritmetică CORECTĂ în forma infixată, fără paranteze, conţinând operatorii binari: +, -, *, /, iar ca operanzi cifre 0-9 (ex: $E = 1\ +\ 2\ *\ 3$). Se cere să se determine forma postfixată $EPost$ a expresiei (ex: $EPost = 1\ 2\ 3\ *\ +$).

Indicaţie: Se va folosi o stivă în care se se vor adăuga operatorii şi o coadă $EPost$ care va conţine în final forma postfixată a expresiei.

Vom putea folosi următorul algoritm.

- Pentru fiecare $e \in E$ (în ordine de la stânga la dreapta)
  1. Dacă e este operand, atunci se adaugă în coada $EPost$.
  2. Dacă e este operator, atunci se scot din stivă operatorii având prioritatea de evaluare mai mare sau egală decât a lui e şi se adaugă în coada $EPost$, după care se adaugă e în stivă.
- Se scot din stivă operatorii rămaşi şi se adaugă în coada $EPost$.
- În final, $EPost$ va conţine forma postfixată a expresiei.

Temă.

Trataţi cazul în care expresia în forma infixată conţine paranteze (ex: $E = (1 + 2) * 3$).

Indicaţie: Ideea/algoritmul de bază este acelaşi ca şi în cazul în care expresia nu ar conţine paranteze. Paranteza deschisă "("se va adăuga în stivă. Va trebui să identificaţi paşii care vor trebui efectuaţi la întâlnirea unei paranteze închise ")".

Probleme

1. Scrieţi 4 proceduri cu timpul de execuţie $\theta(1)$ pentru inserare elemente şi ştergere de elemente la ambele capete ale unei cozi duble (complete).
2. Arătaţi cum se poate implementa o coadă prin 2 stive. Scrieţi în Pseudocod operaţiile cozii folosind doar operaţiile din interfaţa Stivei. Analizaţi timpul de execuţie al operaţiilor cozii.
3. Arătaţi cum se poate implementa o stivă prin 2 cozi. Scrieţi în Pseudocod operaţiile cozii folosind doar operaţiile din interfaą Cozii. Analizaţi timpul de execuţie al operaţiilor stivei.