

TAD Stiva (STACK)

Observații:

1. În limbajul uzual cuvântul “stivă” referă o “grămadă în care elementele constitutive sunt așezate ordonat unele peste altele”.
 - Un element nou se adaugă în stivă deasupra elementului cel mai recent adăugat în stivă.
 - Din stivă se poate accesa și extrage doar elementul cel mai recent introdus.
 - Exemple de stive sunt multiple: stivă de farfurii, stivă de lemne, etc. Tipul de date **Stivă** permite implementarea în aplicații a acestor situații din lumea reală.
2. O *stivă* este o structură liniară de tip listă care restricționează adăugările și ștergerile la un singur capăt (listă LIFO - *Last In First Out*).
3. **Accesul** într-o stivă este *prespecificat* (se poate accesa doar elementul cel mai recent introdus în stivă - din vârful stivei), nu se permite accesul la elemente pe baza poziției. Dintr-o stivă se poate **șterge** elementul CEL MAI RECENT introdus în stivă - cel din vârful stivei.
4. Se poate considera și o capacitate inițială a stivei (număr maxim de elemente pe care le poate include), caz în care dacă numărul efectiv de elemente atinge capacitatea maximă, spunem că avem o *stivă plină*.
 - adăugarea în stiva plină se numește **depășire superioară**.
5. O stivă fără elemente o vom numi *stivă vidă* și o notăm Φ .
 - ștergerea din stiva vidă se numește **depășire inferioară**.
6. O stivă în general nu se iterează.
7. Stivele sunt frecvent utilizate în programare - recursivitate, backtracking iterativ.

Tipul Abstract de Date STIVA:

domeniu:

$$\mathcal{S} = \{s \mid s \text{ este o stivă cu elemente de tip } TElement\}$$

operații (interfața):

- **creeaza**(s)
{creează o stivă vidă}
 $pre : true$
 $post : s \in \mathcal{S}, s = \Phi(\text{stiva vidă})$

- $adauga(s, e)$
 {se adaugă un element în vârful stivei}
 $pre : s \in \mathcal{S}, e \in TElement, s \text{ nu e plină}$
 $post : s' \in \mathcal{S}, s' = s \oplus e, e \text{ va fi cel mai recent element introdus în stivă}$
 $\textcircled{!}$ aruncă excepție dacă stiva e plină
- $sterge(s, e)$
 {se scoate un element din vârful stivei}
 $pre : s \in \mathcal{S}, s \neq \Phi$
 $post : e \in TElement, e \text{ este cel mai recent element introdus în stivă}, s' \in \mathcal{S}, s' = s \ominus e$
 $\textcircled{!}$ aruncă excepție dacă stiva e vidă
- $element(s, e)$
 {se accesează elementul din vârful stivei}
 $pre : s \in \mathcal{S}, s \neq \Phi$
 $post : s' = s, e \in TElement, e \text{ este cel mai recent element introdus în stivă}$
 $\textcircled{!}$ aruncă excepție dacă stiva e vidă
- $vida(s)$
 $pre : s \in \mathcal{S}$
 $post : vida = \begin{cases} adev, & \text{dacă } s = \Phi \\ fals, & \text{dacă } s \neq \Phi \end{cases}$
- $plina(s)$
 $pre : s \in \mathcal{S}$
 $post : plina = \begin{cases} adev, & \text{dacă } s \text{ e plină} \\ fals, & \text{contrar} \end{cases}$
- $distruge(s)$
 {destructor}
 $pre : s \in \mathcal{S}$
 $post : s \text{ a fost 'distrusa' (spațiul de memorie alocat a fost eliberat)}$

Observații

- Stiva nu este potrivită pentru aplicațiile care necesită traversarea ei (nu avem acces direct la elementele din interiorul stivei).
- Afișarea conținutului stivei poate fi realizată folosind o stivă auxiliară (scoatem valorile din stivă punându-le pe o stivă auxiliară, după care se repun pe stiva inițială).

Implementări ale stivelor folosind

- tablouri - vectori (dinamici)
- liste înlănțuite.

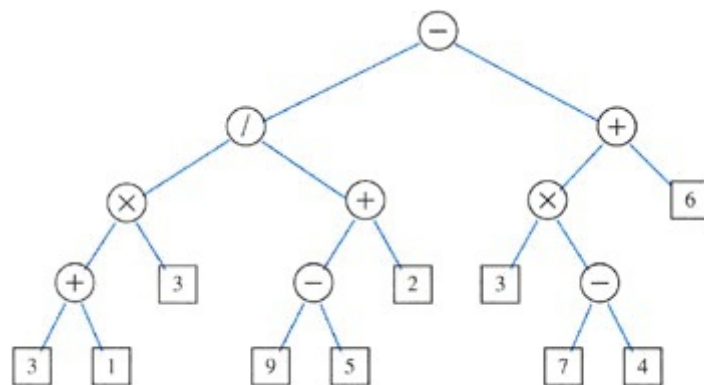


Figura 1: Arbore corespunzător expresiei $((((3+1) \times 3) / ((9-5) + 2)) - ((3 \times (7-4)) + 6))$.

Exemplu

- O expresie aritmetică se poate reprezenta printr-un arbore binar ale cărui noduri terminale sunt asociate cu variabile sau constante și ale cărui noduri interne sunt asociate cu unul dintre operatorii: $+$, $-$, \times , și $/$. (Vezi Figura 1.)
- Fiecare nod dintr-un asemenea arbore are o valoare asociată:
 - Dacă nodul este terminal valoarea sa este cea a variabilei sau constantei asociate.
 - Dacă nodul este neterminal valoarea sa este definită prin aplicarea operației asociate asupra fiilor lui.

Aplicație

Evaluarea unei expresii aritmetice din forma postfixată. Fie $EPost$ o expresie aritmetică CORECTĂ în forma postfixată, conținând operatorii binari: $+$, $-$, \times , $/$, iar ca operanzi cifre 0-9 (ex: $EPost = 1\ 2\ +\ 3\ *\ 4\ /\$). Se cere să se determine valoarea expresiei (ex: valoarea este 2.25).

Indicație: Se va folosi o stivă în care se vor adăuga operanzii. În final, stiva va conține valoarea expresiei.

Vom putea folosi următorul algoritm.

- Pentru fiecare $e \in EPost$ (în ordine de la stânga la dreapta)
 1. Dacă e este operand, atunci se adaugă în stivă.
 2. Dacă e este operator, atunci se scot din stivă doi operanzi (op_1 și op_2), se efectuează operația e între cei doi operanzi ($v = op_2\ e\ op_1$), după care se adaugă v în stivă. **Obs.** Aici s-ar putea depista dacă expresia nu ar fi corectă în forma postfixată (s-ar încerca extragerea unui element dintr-o stivă vidă).
- În presupunerea noastră că expresia în forma postfixată este validă, în final, stiva va conține o singură valoare, care se va extrage din stivă. Această valoare reprezintă valoarea expresiei.

Temă

Se dă un text care conține caractere incluzând paranteze rotunde, paranteze drepte și acolade. Se cere să se verifice dacă în text parantezele se închid corect. De exemplu în textul $a = (2 + b[3]) * 5$; parantezele se închid corect; în textul $a = (b[0] \cdot 1)$; parantezele nu se închid corect. **În aplicație, se vor folosi operațiile din interfața TAD Stivă.**