TAD Stiva (STACK)

Observaţii:

1. În limbajul uzual cuvântul "stivă" referă o "grămadă în care elementele constitutive sunt aşezate ordonat unele peste altele".

   - Un element nou se adaugă în stivă deasupra elementului cel mai recent adăugat în stivă.
   - Din stivă se poate accesa şi 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 aplicaţii a acestor situaţii din lumea reală.

2. O *stivă* este o structură liniară de tip listă care restricţionează adăugările şi ştergerile 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 intodus în stivă - din vârful stivei), nu se permite accesul la elemente pe baza poziţiei. Dintr-o stivă se poate **şterge** elementul CEL MAI RECENT introdus în stivă - cel din vârful stivei.

4. Se poate considera şi o capacitate iniţială 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 numeşte **depăşire superioară**.

5. O stivă fără elemente o vom numi *stivă vidă* şi o notăm Φ.

   - ştergerea din stiva vidă se numeşte **depăşire 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:

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

operaţii (interfaţa):

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

- adauga(s, e)
  {se adaugă un element în vârful stivei}
  $pre:$ $s \in \mathcal{S}, e \in TElement, s$ nu e plină
  $post:$ $s' \in \mathcal{S}, s' = s \oplus e,$ e va fi cel mai recent element introdus în stivă

  @ aruncă excepţie 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 este cel mai recent element introdus în stivă, $s' \in \mathcal{S}, s' = s \ominus e$

  @ aruncă excepţie 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 este cel mai recent element introdus în stivă

  @ aruncă excepţie 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 a fost 'distrusa' (spaţiul de memorie alocat a fost eliberat)

Observaţii

- Stiva nu este potrivită pentru aplicaţiile care necesită traversarea ei (nu avem acces direct la elementele din interiorul stivei).
- Afişarea conţinutului stivei poate fi realizată folosind o stivă auxiliară (scoatem valorile din stivă punându-le pe o stivă auxiliară, după care se repun pe stiva iniţială). Complexitatea timp a sublgoritmului **tiparire** (descris mai jos) este $\theta(n)$, n fiind numărul de elemente din stivă.

```
Subalgoritm tiparire(s)
  {pre: s este o Stivă}
  {post: se tipăresc elementele din Stivă}
  creeaza(sAux) {se creează o stivă auxiliară vidă}
  {se şterg elementele din s şi se adaugă în sAux}
```

```
    CatTimp ¬ vida(s) executa
      sterge(s, e)
      adauga(sAux, e)
    SfCatTimp
    {se şterg elementele din sAux şi se reface s}
    CatTimp ¬ vida(sAux) executa
      sterge(sAux, e)
      @ tipăreşte e
      adauga(s, e)
    SfCatTimp
  SfSubalgoritm
```

Implementări ale stivelor folosind

- tablouri - vectori (dinamici)
- liste înlănţuite.