TAD CoadaCuPrioritati (PRIORITY QUEUE)

Observaţii:

1. O *coadă cu priorităţi* este un container în care fiecare element are asociat o anumită prioritate.
2. Principiul de gestionare a unei cozi cu prioritaţi este "HPF - Highest Priority First".
3. **Accesul** într-o coadă cu priorităţi este *prespecificat* (se poate accesa doar elementul cel mai prioritar - prioritate maximă/minimă). Pentru generalitate, se poate considera o relaţie de ordine $\mathcal{R}$ între priorităţi ($\mathcal{R} : TPrioritate$ x $TPrioritate$, $\mathcal{R}$ relaţie de ordine). În cazul general, vom considera că se accesează elementul 'cel mai prioritar' în raport cu relaţia de ordine $\mathcal{R}$ (de ex. dacă se doreşte accesul la elementul de prioritate maximă, atunci $\mathcal{R} =$ " $\geq$"). Dintr-o coadă cu priorităţi se **şterge** elementul 'cel mai prioritar' (în raport cu relaţia de ordine $\mathcal{R}$). **Inserarea** unui element se va face astfe încât să se păstreze relaţia de ordine între priorităţi.
4. Se poate considera şi o capacitate iniţială a cozii cu priorităţi (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 cu priorităţi plină*.
5. Cozile cu priorităţi sunt mai simple decât dicţionarele (se restricţionează accesul).
6. O coadă cu priorităţi în general nu se iterează.
7. Aplicaţii
   - simularea unor sisteme complexe (aeroport) - sunt mai multe evenimente care au loc la un anumit moment de timp **t**. Simularea se va face printr-o coadă cu priorităţi (**t** este prioritatea) - accesul este la elementul având prioritate *minimă*.
   - probleme de concurenţă a proceselor.

Tipul Abstract de Date COADA CU PRIORITATI:

domeniu:

$\mathcal{CP} = \{cp \mid cp$ este o coadă cu priorităţi cu elemente (e, p) de tip $TElement$ x $TPrioritate\}$

operaţii:

- creeaza$(cp, \mathcal{R})$
  {creează o coadă cu priorităţi vidă}
  $pre:$ $\mathcal{R} : TPrioritate$ x $TPrioritate, \mathcal{R}$ relaţie de ordine
  $post:$ $cp \in \mathcal{CP}, cp = \Phi$

- adauga(cp, e, p)
  {se adaugă un element în coada cu priorităţi}
  $pre:$ $cp \in \mathcal{CP}, e \in TElement, p \in TPrioritate$
  $post:$ $cp' \in \mathcal{CP}, cp' = cp \oplus (e, p)$

- sterge(cp, e, p)
  {se sterge elementul 'cel mai prioritar'}
  $pre:$ $cp \in \mathcal{CP}, cp \neq \Phi$
  $post:$ $e \in TElement, p \in TPrioritate,$ e este 'cel mai prioritar' element din cp
  p e prioritatea sa, $cp' \in \mathcal{CP}, cp' = cp \ominus (e, p)$

  @ aruncă excepţie dacă coada cu priorităţi e vidă

- element(cp, e, p)
  {se accesează elementul 'cel mai prioritar'}
  $pre:$ $cp \in \mathcal{CP}, cp \neq \Phi$
  $post:$ $cp' = cp, e \in TElement, p \in TPrioritate$
  e este 'cel mai prioritar' element din cp, p e prioritatea sa

  @ aruncă excepţie dacă coada cu priorităţi e vidă

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

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

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

Observaţii

- CP nu este potrivită pentru aplicaţiile care necesită traversarea ei (nu avem acces direct la elementele din interiorul cozii).

- Afişarea conţinutului CP poate fi realizată (ca şi la o **coadă** simplă) folosind o CP auxiliară. Complexitatea timp a sublgoritmului **tiparire** (descris mai jos) este $\theta(n)$, n fiind numărul de elemente din CP.

`Subalgoritm tiparire`(cp)
{*pre: cp este o CP*}
{*post: se tipăresc elementele din cp*}

```
  creeaza(cpAux) {se creează o CP auxiliară vidă}
  {se şterg elementele din cp şi se adaugă în cpAux}
  CatTimp ¬ vida(cp) executa
    sterge(cp, e)
    @ tipăreşte e
    adauga(cpAux, e)
  SfCatTimp
  {se şterg elementele din cpAux şi se reface cp}
  CatTimp ¬ vida(cpAux) executa
    sterge(cpAux, e)
    adauga(cp, e)
  SfCatTimp
SfSubalgoritm
```

Implementări ale cozilor cu priorităţi folosind

- o listă sortată (ordonată) în raport cu prioritatea elementelor
  - reprezentare secvenţială pe tablou (vector dinamic) .
  - reprezentare înlănţuită (simplu/dublu, reprezentare înlănţuiri folosind alocare dinamică/alocare statică).
- **ansamblu (heap)** (va fi discutat în Cursul 7).

Observaţie:

- *Ansamblul* este cea mai potrivită SD pentru implementarea unei CP, deoarece oferă o complexitate $O(log_2 n)$ pentru operaţiile specifice (adăugare, ştergere) şi $\theta(1)$ pentru operaţia de accesare.

Aplicaţie

Acordarea burselor pentru studenţi. Să se simuleze procesul de acordare a burselor pentru studenţi. Sunt n studenţi, se acordă m burse. Bursele se acordă în ordine descrescătoare a mediilor.