SDA - Seminar - TAD Colecție

- **Cuprins**:
  - Convenții
  - TAD Colecție – definire și specificare
  - TAD Iterator – specificare
  - Exemplificare reprezentare
  - Implementare Python

Convenții

- Algoritmii vor fi descriși în **Pseudocod**
- Elementele din containere vor fi elemente generale/abstracte
  - **TElement**
    - Operație de atribuire: $e_1 \leftarrow e_2$
    - Verificarea egalității: $e_1 = e_2$
  - **TComparabil**
    - Pentru containere ordonate
    - Permite inclusiv aplicarea unor relații de ordine: <, > etc.

TAD Colecție

Definire

- Spre deosebire de cazul mulțimii, elementele nu sunt neapărat distincte
- Ca și în cazul mulțimii, ordinea elementelor este irelevantă
  - Nu există poziții într-o colecție. Așadar, operațiile colecției nu primesc poziții ca parametri și nu returnează poziții.
  - Elementele nu sunt stocate în ordinea adăugării (cel puțin, nu avem această garanție)

    De exemplu, dacă adăugăm într-o colecție elementele 1,3,2,6,2,5,2, la afișarea conținutului colecției i, orice ordine a elementelor e posibilă:
    - 1,2,2,2,3,6,5
    - 1,3,2,6,2,5,2
    - 1,5,6,2,3,2,2

    etc…

Specificare

1) **Definirea domeniului:**

$\mathcal{C}$ = {c | c este o colecție cu elemente de tip TElement}

2) **Specificarea interfeței:**

creeaza(c)

pre: adevărat
post: c ∈ $\mathcal{C}$, c este colecția vidă (sau dim(c)=0)

distruge(c)

pre: c ∈ $\mathcal{C}$

post: colecția c a fost distrusă

adauga(c,e)

pre: c ∈ $\mathcal{C}$, e ∈ TElement
post:c' ∈ $\mathcal{C}$, c'=c ∪ {e} (sau c'= c ⊕ {e})

sterge(c,e)

pre: c ∈ $\mathcal{C}$, e ∈ TElement
post: c' ∈ $\mathcal{C}$, c'=c \ {e} (sau c'= c ⊖{e}; **OBS**: se șterge o singură apariție a elementului e)

cauta(c,e)

pre: c ∈ $\mathcal{C}$, e ∈ TElement
post:

$$\text{cauta} \leftarrow \begin{cases} \textit{adevărat}, \text{ dacă e} \in \text{c} \\ \textit{fals}, \text{ altfel} \end{cases}$$

dim(c)

pre: c ∈ $\mathcal{C}$
post: dim ← numărul total de elemente din c

iterator(c,i)

pre: c ∈ $\mathcal{C}$
post:i ∈ $\mathcal{I}$, i este iterator pe c și i referă un prim element din c

TAD Iterator

Specificare

1) **Definirea domeniului:**

$\mathcal{I}$ = {i|i este iterator pe c ∈ $\boldsymbol{C}$}

2) **Specificarea interfeței:**

creeaza(i,c)

pre: c ∈ $\boldsymbol{C}$

post: i ∈ $\mathcal{I}$, i este iterator pe c și i referă un prim element din c

valid(i)

pre: i ∈$\mathcal{I}$

post:

valid ← { *adevărat*, dacă elementul curent referit de i este valid; *fals*, altfel

urmator(i)

pre: i ∈$\mathcal{I}$

post: i' ∈ $\mathcal{I}$, i' referă următorul element din colecția iterată față de cel referit de i

element(i, e)

pre: i ∈ $\mathcal{I}$

post: e ∈ TElement, e este elementul curent referit de iterator

Exemplificare reprezentare:

O primă variantă de reprezentare (R1):

- Ca tablou unidimensional (vector) de elemente care se pot repeta
- În acest caz, iteratorul conține indexul elementului curent

Exemplu:

1	3	2	6	2	5	2

<u>O a doua variantă de reprezentare (R2):</u>

- Ca vector de perechi (element,frecventa),elementele din perechi fiind distincte
- În acest caz, nu mai este suficient ca în reprezentarea iteratorului să avem doar indexul curent. Este necesară și frecvența curentă pentru elementul de la indexul curent.

Exemplu:

(1, 1)	(3, 1)	(2, 3)	(6, 1)	(5, 1)

Implementare

- a se vedea fișierele **R1.py** (prima reprezentare) și **R2.py** (a doua reprezentare)