Colecţia

COLLECTION, BAG, MULTI-SET

Observații

- **Colecţie** ("**bag**") este un container, o grupare finită de elemente.
- Într-o colecţie elementele nu sunt distincte (nu există o singură instanţă a unui element).
  - Din această cauză colecţia mai e cunoscută sub numele de **multi-mulţime** ("**multi-set**").
  - Operaţii specifice pe o colecţie sunt: adăugarea, ştergerea, căutarea unui element într-o colecţie.
    - Ca urmare tipul elementelor din colecţie, **TElement** ar trebui să suporte cel puţin operaţiile de: atribuire (⟵) şi testarea egalităţii (=).
- Caracteristică a colecţiei - nu contează ordinea elementelor.
  - Spre exemplu, o colecţie de numere întregi ar putea fi: ***c***={1, 2, 3, 1, 3, 2, 4, 2, 2}.

În continuare, vom prezenta specificaţia Tipul Abstract de Date **Colecţie.**

domeniu

$\mathcal{Col}$={**col | col** este o colecţie cu elemente de tip **TElement**}

operaţii (**interfaţa** TAD-ului Colecţie)

creează(c)

pre: -

post:c∈ $\mathcal{Col}$, c este colecţia vidă (fără elemente)

adaugă(c, e)

pre: c∈ $\mathcal{Col}$, e∈ **TElement**

post: c'∈ $\mathcal{Col}$, c'=c+{e}

{s-a adăugat elementul în colecţie}

şterge(c, e)

pre: c∈ $\mathcal{Col}$, e∈ **TElement**

post: c'∈ $\mathcal{Col}$, c'=c-{e}

{s-a eliminat o apariţie a elementului din colecţie}

caută(c, e)

pre: c∈ $\mathcal{Col}$, e∈ **TElement**

post: ***caută*** **=** adevărat dacă e∈ c

fals în caz contrar

dim(c)

pre: c∈ $\mathcal{Col}$,

post: ***dim*** **=** dimensiunea colecţiei c (numărul de elemente) ∈ $\mathcal{N}$

vidă(c)

pre: c∈ $\mathcal{Col}$

post: ***vidă*****=** adevărat în cazul în care c e colecţia vidă
fals în caz contrar

iterator(c, i)

pre: c∈ $\mathcal{Col}$

post: i∈ $\mathcal{I}$, i este un iterator pe colecţia c

distruge(c)

pre: c∈ $\mathcal{Col}$

post: colecţia c a fost 'distrusă' (spaţiul de memorie alocat a fost eliberat)

Menţionăm că pot fi definite în interfaţa Tipului Abstract de Date Colecţie şi operaţii specifice cum ar fi: reuniunea, intersecţia, diferenţa a două colecţii.

Deoarece colecţia are o operaţie care furnizează un iterator pe elementele sale, subalgoritmul care va tipări elementele unei colecţii ***c*** poate fi descris sub forma:

Subalgoritmul *tipărire*(c) **este**
{pre: c este o colecţie}
{post: se tipăresc elementele colecţiei}
iterator(c,i) {colecţia îşi construieşte iteratorul}
CâtTimp valid(i) **execută** {cât timp iteratorul e valid}
element(i, e) {se obţine elementul curent din iteraţie}
@ tipăreşte e {se tipăreşte elementul curent}
următor(i) {se deplasează iteratorul}
SfCâtTimp
SfTipărire

Complexitatea-timp a subalgoritmului de tipărire este θ (|c|), unde prin |c| am notat dimensiunea colecţiei c.

Ca şi modalităţi de reprezentarea ale unei colecţii, avem cel puţin două astfel de posibilităţi:

- se reprezintă toate elementele colecţiei : 1, 2, 1, 4, 3, 1, 4, 2, 5;
- se reprezintă colecţia sub forma unor perechi de forma (e_1, f_1), (e_2, f_2),..,(e_n, f_n), unde $e_1, e_2,...,e_n$ reprezintă elementele distincte din colecţie, iar $f_1,f_2,...,f_n$ reprezintă frecvenţele de apariţie (numărul de apariţii în colecţie) a elementelor corespunzătoare: spre exemplu colecţia anterioară s-ar reprezenta sub forma perechilor (1, 3), (2, 2), (4, 2), (3, 1), (5, 1).

Modalităţi de implementare ale colecţiilor ar putea fi folosind:

- tablouri (dinamice);
- liste înlănţuite;
- tabele de dispersie;
- arbori binari.

Mulţimea (SET)

O **mulţime** ("set") este un container cu ajutorul căruia se poate reprezenta o colecţie finită de elemente distincte. Altfel spus, într-o mulţime elementele nu se pot repeta, există o singură instanţă a unui element. O altă caracteristică a mulţimii este faptul că într-o mulţime nu contează ordinea elementelor.

Mulţimea are toate operaţiile specifice **Colecţiei**, cu observaţia că operaţia adăugare într-o mulţime are specificaţie diferită faţă de operaţia de adăugare într-o colecţie (într-o mulţime elementele trebuie să fie distincte).

Tipul elementelor din mulţime, **TElement**, ca şi într-o colecţie, dealtfel, suportă cel puţin operaţiile de: atribuire (⟵) şi testarea egalităţii (=).

Spre exemplu, o mulţime de numere întregi ar putea fi: ***m***={1, 2, 3, 5, 4}.

Caracterul finit al unei mulţimi ne permite (totuşi) indexarea elementelor sale, ceea ce face ca la nivelul reprezentării interne, mulţimea **M** să poată fi asimilată cu un vector m_1, m_2,..., m_n (chiar dacă ordinea elementelor dintr-o mulţime nu este esenţială).

Pentru a putea preciza modul în care se vor efectua operaţiile pe mulţimi, vom defini structura de ***submulţime*** . Aceasta se poate realiza cu ajutorul unui vector format din valorile funcţiei caracteristice asociate submulţimii.

Dacă **M** este o mulţime, atunci submulţimea **S⊆M** va avea asociat vectorul $\mathbf{V_S}$=(s_1, s_2,..., s_n) unde

$$s_i = \begin{cases} 1, & daca\ m_i \in S \\ 0, & daca\ m_i \notin S \end{cases}$$

Operaţiile pe submulţimi pot fi acum definite prin intremediul operaţiilor pe vectorii caracteristici asociaţi.

Fie **S1, S2 ⊆M** două submulţimi ale mulţimii **M**. Atunci:

a) **S1 ∪ S2 (reuniunea celor două submulţimi)** va fi caracterizată de vectorul **V** obţinut din $\mathbf{V_{S1}}$ şi $\mathbf{V_{S2}}$ efectuând operaţia logică "∨" (sau) element cu element

∨	0	1
0	0	1
1	1	1

b) S1 $\cap$ S2 (intersecţia celor două submulţimi) va fi caracterizată de vectorul **V** obţinut din $\mathbf{V_{S1}}$ şi $\mathbf{V_{S2}}$ efectuând operaţia logică "$\wedge$" (sau) element cu element

$\wedge$	0	1
0	0	0
1	0	1

Ca urmare, orice alte operaţii cu submulţimile unei mulţimi pot fi imaginate ca operaţii logice asupra vectorilor ataşaţi.

În continuare, vom prezenta specificaţia Tipul Abstract de Date **Mulţime.**

domeniu

$\mathcal{M}$={**m** | **m** este o mulţime cu elemente de tip **TElement**}

operaţii (**interfaţa** TAD-ului Mulţime)

creează(m)

pre: -

post:m$\in \mathcal{M}$, m este mulţimea vidă (fără elemente)

adaugă(m, e)

pre: m$\in \mathcal{M}$, e$\in$**TElement**

post: m'$\in \mathcal{M}$, m'=m$\cup${e}

{e se "reuneşte" la mulţime, adică se va adăuga numai dacă e nu mai apare în mulţime}

şterge(m, e)

pre: m$\in \mathcal{M}$, e$\in$**TElement**

post: m'$\in \mathcal{M}$, m'=m-{e}

{se şterge e din m}

caută(m, e)

pre: m$\in \mathcal{M}$, e$\in$**TElement**

post: ***cauta*=** adevărat dacă e$\in$m
fals în caz contrar

dim(m)

pre: m$\in \mathcal{M}$

post: ***dim*=** dimensiunea mulţimii m (numărul de elemente) $\in \mathcal{N}$

vidă(m)

pre: m$\in \mathcal{M}$

post: ***vida*=** adevărat în cazul în care m e mulţimea vidă
fals în caz contrar

iterator(m, i)

pre: m$\in \mathcal{M}$

post: i$\in \mathcal{I}$, i este un iterator pe mulţimea m

distruge(m)

pre: m$\in \mathcal{M}$

post: mulţimea m a fost 'distrusă' (spaţiul de memorie alocat a fost eliberat)

Accesarea elementelor mulţimii se va face în aceeaşi manieră ca la colecţie, folosind iteratorul pe care-l oferă mulţimea.

Modalităţi de implementare ale mulţimilor:

- tablouri (dinamice);
- vectori booleeni (de biţi);
- liste înlănţuite;
- tabele de dispersie;
- arbori binari.

TEMA. Implementaţi operaţiile specifice TAD Colecţie, Mulţime folosind SD menţionate. Studiaţi complexitatea operaţiilor în funcţie de SD aleasă pentru implementare.