

## Mulțimea (SET)

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

**Mulțimea** are toate operațiile specifice **Colecției**, cu observația că operația adăugare într-o mulțime are specificație diferită față de operația de adăugare într-o colecție (într-o mulțime elementele trebuie să fie distincte).

Tipul elementelor din mulțime, **TElement**, ca și într-o colecție, suportă cel puțin operațiile de: atribuire ( $\leftarrow$ ) și testarea egalității ( $=$ ).

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

Caracterul finit al unei mulțimi ne permite (totuși) indexarea elementelor sale, ceea ce face ca la nivelul reprezentării interne, mulțimea **M** să poată fi asimilată cu un vector  $m_1, m_2, \dots, m_n$  (chiar dacă ordinea elementelor dintr-o mulțime nu este esențială).

Pentru a putea preciza modul în care se vor efectua operațiile pe mulțimi, vom defini structura de **submulțime**. Aceasta se poate realiza cu ajutorul unui vector format din valorile funcției caracteristice asociate submulțimii.

Dacă **M** este o mulțime, atunci submulțimea  $S \subseteq M$  va avea asociat vectorul  $V_S = (s_1, s_2, \dots, s_n)$  unde

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

De exemplu, dacă  $M = \{110, 200, 318, 400\}$ , atunci submulțimea  $S = \{200, 400\}$  a lui **M** se va reprezenta sub forma vectorului **(0, 1, 0, 1)** sau **(false, true, false, true)**.

Operațiile pe submulțimi pot fi acum definite prin intermediul operațiilor pe vectorii caracteristici asociați.

Fie  $S_1, S_2 \subseteq M$  două submulțimi ale mulțimii **M**. Atunci:

- a)  $S_1 \cup S_2$  (**reuniunea celor două submulțimi**) va fi caracterizată de vectorul **V** obținut din  $V_{S_1}$  și  $V_{S_2}$  efectuând operația logică " $\vee$ " (sau) element cu element

$\vee$	0	1
0	0	1
1	1	1

- b)  $S_1 \cap S_2$  (**intersecția celor două submulțimi**) va fi caracterizată de vectorul **V** obținut din  $V_{S_1}$  și  $V_{S_2}$  efectuând operația logică " $\wedge$ " (sau) element cu element

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

Ca urmare, orice alte operații cu submulțimile unei mulțimi pot fi imaginate ca operații logice asupra vectorilor atașați.

În continuare, vom prezenta specificația Tipul Abstract de Date **Mulțime**.

**domeniu**

$\mathcal{M} = \{m \mid m \text{ este o mulțime cu elemente de tip } TElement\}$

**operații (interfața TAD-ului Mulțime)**

*creează*(m)

**pre:** -

**post:**  $m \in \mathcal{M}$ , m este mulțimea 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 "reunește" la mulțime, adică se va adăuga numai dacă e nu mai apare în mulțime.}  
{se poate returna adevărat dacă elementul a fost adăugat}

*șterge*(m, e)

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

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

{se șterge e din m}  
{se poate returna adevărat dacă elementul a fost șters}

*caută*(m, e)

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

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

*dim*(m)

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

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

*vidă*(m)

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

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

*iterator*(m, i)

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

**post:**  $i \in I$ , i este un iterator pe mulțimea m

*distruge*(m)

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

**post:** mulțimea  $m$  a fost 'distrusă' (spațiul de memorie alocat a fost eliberat)

Accesarea elementelor mulțimii se va face în aceeași manieră ca la colecție, folosind iteratorul pe care-l oferă mulțimea.

**Observație.** În cazul în care elementele din mulțime sunt de tip **TComparabil**, elementele pot fi memorate în ordine (în raport cu o anumită relație de ordine, de ex.  $\leq$ ), pentru a reduce complexitatea timp a unor operații.

Modalități de implementare ale mulțimilor:

- tablouri (dinamice);
- vectori booleeni (de biți);
- liste înlanțuite;
- tabele de dispersie;
- arbori binari (de căutare echilibrați).