Colecția

COLLECTION, BAG, MULTI-SET

Observații

- Colecție ("bag") este un container, o grupare finită de elemente.
- Într-o colecție elementele nu sunt distincte (nu există o singură instanță a unui element).
 - Din această cauză colecția mai e cunoscută sub numele de multi-multime ("multi-set").
 - Operații specifice pe o colecție sunt: adăugarea, ștergerea, căutarea unui element într-o colecție.
 - o Ca urmare tipul elementelor din colecție, **TElement** ar trebui să suporte cel puțin operațiile de: atribuire (←) și testarea egalității (=).
- Caracteristică a colecției nu contează ordinea elementelor.
 - Spre exemplu, o colecție de numere întregi ar putea fi: $c = \{1, 2, 3, 1, 3, 2, 4, 2, 2\}$.

În continuare, vom prezenta specificația Tipul Abstract de Date Colecție (interfața minimală).

domeniu

```
Col={col | col | este o colecție cu elemente de tip TElement}
operații (interfața TAD-ului Colecție)
       creează(c)
               pre: -
               post:c∈ Col, c este colectia vidă (fără elemente)
       adaugă(c, e)
               pre: c \in CoL e \in TElement
               post: c' \in CoL c' = c + \{e\}
                {s-a adăugat elementul în colecție}
       sterge(c, e)
               pre: c \in CoL e \in TElement
               post: c' \in Col, c' = c - \{e\}
                {s-a eliminat o apariție a elementului din colecție}
                {se poate returna adevărat dacă elementul a fost șters}
       caută(c, e)
               pre: c \in CoL e \in TElement
               post: caută = adevărat
                                               dacă e∈c
```

```
fals
                                        în caz contrar
dim(c)
       pre: c∈ Col
       post: dim = dimensiunea colecției c (numărul de elemente) \in \mathcal{N}
vidă(c)
       pre: c \in Col
        post: vidă= adevărat
                                        în cazul în care c e colecția vidă
                    fals
                                        în caz contrar
iterator(c, i)
       pre: c \in Col
       post: i \in I, i este un iterator pe colecția c
distruge(c)
       pre: c \in Col
       post: colecția c a fost 'distrusă' (spațiul de memorie alocat a fost eliberat)
```

Menționăm că pot fi definite în interfața Tipului Abstract de Date Colecție și operații specifice cum ar fi: reuniunea, intersecția, diferența a două colecții.

Deoarece colecția are o operație care furnizează un iterator pe elementele sale, subalgoritmul care va tipări elementele unei colecții c poate fi descris sub forma:

```
Subalgoritmul tipărire(c) este

{pre: c este o colecție}

{post: se tipăresc elementele colecției}

iterator(c,i) {colecția își construiește iteratorul}

CâtTimp valid(i) execută {cât timp iteratorul e valid}

element(i, e) {se obține elementul curent din iterație}

@ tipărește e {se tipărește elementul curent}

următor(i) {se deplasează iteratorul}

SfCâtTimp

SfTipărire
```

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

Ca și modalități de reprezentarea ale unei colecții, avem cel puțin următoarele posibilități:

- se reprezintă toate elementele colecției : 1, 2, 1, 4, 3, 1, 4, 2, 5;
- se reprezintă colecția sub forma unor perechi de forma (e₁, f₁), (e₂, f₂),...,(e_n, f_n), unde e₁, e₂,...,e_n reprezintă elementele distincte din colecție, iar f₁,f₂,...,f_n reprezintă frecvențele de apariție (numărul

de apariții în colecție) a elementelor corespunzătoare: spre exemplu colecția anterioară s-ar reprezenta sub forma perechilor (1, 3), (2, 2), (4, 2), (3, 1), (5, 1).

<u>Observație.</u> În cazul în care elementele din colecție 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 colecțiilor ar putea fi folosind:

- tablouri (dinamice);
- liste înlănțuite;
- tabele de dispersie;
- arbori binari (de căutare echilibrați).

Implementări în biblioteci predefinite

- Java
 - o interfața Collection
 - clase care implementează interfața
 - Vector, ArrayList, Stack, etc
- STL
 - o unordered multiset
 - implementare tabelă de dispersie (complexitate medie O(1))

În directorul TAD Colecție (curs 3) găsiți implementarea parțială, în limbajul C++, a containerului **Colecție**, reprezentare secvențială pe vector dinamic (se memorează toate elementele colecției în vector).