**DOCUMENTATIE**

(**TAD Multime – implementare folosind o tabela de dispersie / rezolvare coliziuni prin liste independente.**)

**Nume și prenume**: Csibi Flavius-Petru

**Grupa**: 212

***Specificare și interfață TAD***

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

**creează**(m)

**pre**: -

**post**:m**∈*M***, m este mulţimea vidă (fără elemente)

**adaugă**(m, e)

**pre**: m**∈*M***, e∈**TElement**

**post**: m'**∈*M***, m'=m∪{e}

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

**şterge**(m, e)

**pre**: m**∈M**, e**∈TElement**

**post**: m'**∈M**, m'=m-{e}

{se şterge e din m}

**caută**(m, e)

**pre**: m**∈M**, e∈**TElement**

**post**: cauta= adevărat dacă e∈m

fals în caz contrar

**dim**(m)

**pre**: m∈M

**post: dim**= dimensiunea mulţimii m (numărul de elemente) **∈N**

**vidă**(m)

**pre**: m**∈M**

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

fals în caz contrar

**iterator**(m, i)

**pre**: m**∈M**

**post**: i**∈I**, i este un iterator pe mulţimea m

**distruge**(m)

**pre**: m**∈M**

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

**IteratorMultime** = {**i** | **i** este un iterator pe o mulțime cu elemente de tip TElement}

***creeaza*(i, c)**

**pre:** m **∈** **Multime**

**post:** i**∈IteratorMultime,** s-a creat iteratorul i pe multimea m (elementul

*curent* din iterator referă ‘primul’ element din multime)

***element*(i, e)**

**pre:** i **∈** **IteratorMultime**

**post:** e **∈** TElement, e este elementul curent din iterație

***valid*(i)**

**pre:** i **∈** **IteratorMultime**

**post: *valid*** = adevărat, dacă *curent* referă o poziție validă din multime

fals, altfel

***următor*(i)**

**pre:** i **∈** **IteratorMultime**

**post:** *curent* referă următorul element din multime

***Proiectare și implementare TAD***

*-implementare folosind o tabela de dispersie / rezolvare coliziuni prin liste independente.*

**Nod**

element: ***TElement*** {elementul de tip ***TElement***}

urmator: ***Intreg*** {poziția din vector a nodului următor}

**Multime**

multime: ***Nod[]*** {vector cu elementele din multime, stocate in ***Nod***}

prim: ***Intreg*** {poziția primului element}

primLiber: ***Intreg*** {prima poziție liberă din vector}

ultim: ***Intreg*** {poziția ultimului element}

dim: ***Intreg*** {numărul elementelor din multime}

**IteratorMultime**

multime: ***Multime*** {multimea iterată}

curent: ***Intreg*** {poziția elementului curent din iterație}

***Multime***

**subalgoritmul** *creează*(mul) **este**

max = 13;

ht ← aloca ↑Nod[max];

pentru i =1 , max

ht[i] ← NILL

sfPentru

**sfCreează**

**subalgoritmul** *adaugă*(mul, e) **este**

pos ← hashFunction(element)

↑Nod crt = ht[pos]

**catTimp**(crt ≠ NILL && (element ≠ [crt].element))

crt = [crt].urmator

**sfCatTimp**

**daca** (crt ≠ NILL)

**return**

**altfel**

↑Nod n ← **aloca** Nod

[n].element ← element;

[n]urmator ← ht[pos]

ht[pos] ← n;

**sfDaca**

dim ← dim +1

**sfAdaugă**

**subalgoritmul** *șterge*(mul, e) este

pos ← hashFunction(element); {poz: intreg}

↑Nod crt ← ht[pos]

daca ht[pos] = NULL

return

**daca** [ht[pos]].element = element

↑Nod deSters ← ht[pos];

ht[pos] ←[ ht[pos]].urmator

sterge deSters

dim ← dim-1

return

**sfDaca**

↑Nod elem ← ht[pos]

**catTimp** [elem].urmator ≠ NILL si [[elem].urmator].element ≠ element

elem ← [elem].urmator

**sfCatTimp**

**daca** [elem].urmator ≠ NILL

↑Nod deSters ← [elem].urmator

[elem].urmator ← [[elem].urmator].urmator;

sterge deSters

dim ← dim-1

**sfDaca**

**sfȘterge**

**funcția** *dimensiune*(mul) **este**

***dimensiune*** ← ***dim***

**sfDimensiune**

**funcția** *vidă*(mul) **este**

***vidă*** ← mul.dim = 0

**sfVidă**

**subalgoritmul** *iterator*(mul, i) **este**

creează(i, mul)

**sfIterator**

**funcția** *caută*(mul, e) **este**

pos ← hashFunction(element); {post**: Intreg**; gasit: **Bool**}

gasit ← fals;

↑Nod crt ← ht[pos]

**cattimp**(crt ≠ NILL) si (¬gasit)

**daca** crt ≠ NILL

gasit ← **adevarat**

altfel

crt ← [crt].urmator;

**sfDaca**;

sfCatTimp

cauta ← gasit

**sfCaută**

*IteratorMultime*

**subalgoritmul** *creează*(i, m) este

multime ← mul;

crt ← [mul].prim;

**sfCreează**

**subalgoritmul** *element*(i, e) **este**

e ← [crt].element;

**sfElement**

**funcția** *valid*(i) **este**

**valid**←crt ≠ NILL

**sfValid**

**subalgoritmul** *următor*(i) **este**

i.crt ← [crt].urmator;

**sfUrmător**

***Aplicația (enunț și pseudocod)***

Un candidat la primaria unei localitati vrea sa vada care sunt sansele sale la castigare, iar pentru aceasta toti oamenii care vor sa voteze pentru el ii trimit acestuia CNP-ul,care este un cod unic, fiecare persoana poate aparea doar o data. Pentru aceasta el are nevoie de o aplicatie sa ii gestioneze aceasta baza de date.

*ContainerRepository*

**subalgoritmul** *adaugă*(mul , cnt) **este**

*{cnt:* ***Container;*** *mul:* ***Multime****}*

mul.adaugă(cnt)

**sfAdaugă**

**subalgoritmul** *șterge*(mul , cnt) **este**

*{cnt:* ***Container;*** *mul:* ***Multime****}*

mul.șterge(cnt)

**sfȘterge**

**functia** *caută*(mul , cnt) **este**

*{cnt:* ***Container;*** *mul:* ***Multime****}*

***caută*** ← mul.caută(cnt)

**sfCaută**

**functia** *dimensiune*(mul) **este**

*{mul:* ***Multime****}*

***dimensiune*** ← mul.dimensiune(mul)

**sfDimensiune**

*ConsoleUI*

**Subalgoritm** afiseazaContainere() {

{ctrl: ContainerController; it:IteratorMultime}

**daca** [ctrl].size() = 0

**Tipărește** “Baza de date este goala.”

**altfel**

it ← [[ctrl].getMul()].iterator()

**catTimp** it.valid()

**tipărește** it.element.getCnp()

it.urmator()

**sfCâtTimp**

sfDacă

**sfAfiseazaContainere**