SDA – Seminar 4
Multidicționar ordonat (MDO)

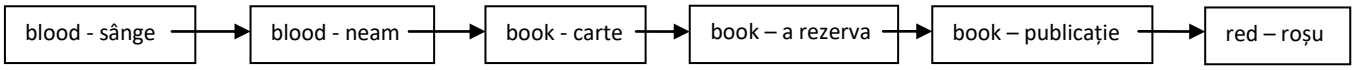
- Dicționar – conține perechi <cheie, valoare>. Cheile sunt unice, fiecare are o singură valoare asociată.
- Multidicționar – o cheie are mai multe valori asociate (listă de valori)
- Multidicționar ordonat – cheile sunt într-o anumită ordine R și sunt memorate în ordine

Interfată MDO

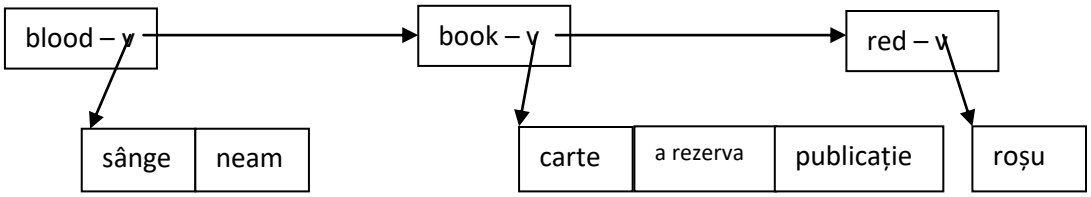
Problema: Să se implementeze TAD MDO – reprezentare simplu înlănțuită cu alocare dinamică

Ex. multidicționar cu traducerile cuvintelor din engleză în română:
- book – carte, a rezerva, publicație
- red – roșu
- blood – sânge, neam

Reprezentare 1: Listă înlănțuită de perechi <cheie, valoare>. Pot fi mai multe noduri cu o anumită cheie, ele vin unul după altul.



Reprezentare 2: Listă înlănțuită de perechi <cheie, listă de valori>. Cheile sunt unice și ordonate.



<u>TElement:</u>
c: TCheie
v: TListă

<u>NodT:</u>
e: TElement
urm: ↑NodT

<u>MDO:</u>
prim: ↑NodT
R: relație

$$R(c_1, c_2) = \begin{cases} adevărat, dacă\ "c_1 \leq c_2"\ (c_1 vine\ înaintea\ lui\ c_2) \\ fals, altfel \end{cases}$$

<u>Iterator:</u>
Vom reține:
- o referință spre nodul curent din MDO
- iterator pe lista de valori asociată nodului curent
- MDO-ul

IteratorMDO:
mdo: MDO
curent: ↑NodT
itL: IteratorListă

Operații iterator: creeaza, valid, următor, element (returnează pereche <cheie, valoare>)

Afișarea elementelor dintr-un MDO folosind iteratorul:

```
Subalg tipărire(mdo):
	iterator(mdo, i)
	câttimp valid(i) execută:
		element(i, <c,v>)
		@tipărește c și v
		următor(i)
	sf_câttimp
sf_subalg
```

Subalgoritmul este la fel indiferent de reprezentarea iteratorului sau a dicționarului!

Operații iterator

```
Subalg creează (it, mdo):
	it.mdo ← mdo
	it.curent ← mdo.prim
	dacă it.curent ≠ NIL atunci:
		iterator([it.mdo.prim].e.l, it.itL)
	sf_dacă
sf_subalg
Complexitate: θ(1)


Subalg element(it, e): // unde e este o pereche, c, v
	c ← [it.curent].e.c
	element(it.itL, v)
	e ← <c,v>
sf_subalg
Complexitate: θ(1)


Funcția valid(it):
	Dacă it.curent ≠ NIL atunci
		valid ← adevărat
	altfel
		valid ← fals
sf_funcție
Complexitate: θ(1)
```

```
Subalg următor(it):
	urmator(it.itL)
	dacă l valid(it.itL) atunci
		it.curent ← [it.curent].urm
		dacă it.curent ≠ NIL atunci
			iterator ([it.curent].e.l, it.itL)
		sf_dacă
	sf_dacă
sf_subalg
Complexitate: θ(1)
```

Operații multidicționar ordonat

La complexități:

n – nr de chei distincte

mdo – nr total de elemente

```
subalg creează(mdo, R):
	mdo.R ← R
	mdo.prim ← NIL
sf_subalg
Complexitate: θ(1)


subalg distruge(mdo):
	câttimp mdo.prim ≠ NIL execută
		p ← mdo.prim
		mdo.prim ← [mdo.prim].urm
		distruge([p].e.l)
		dealocă(p)
	sf_câttimp
sf_subalg
Complexitate: θ(mdo) (sau θ(n) – dacă listele de valori pot fi dealocate in θ(1))


subalgoritm cautNod(mdo, c, nodC, nodAnt):
	aux ← mdo.prim
	prev ← NIL
	găsit ← fals
	câttimp aux ≠ NIL și mdo.R([aux].e.c, c) și l găsit execută
		dacă [aux].e.c = c atunci
			găsit ← adevărat
		altfel
			prev ← aux
			aux ← [aux].urm
		sf_dacă
	sf_câttimp
	dacă găsit atunci
		nodC ← aux
		nodAnt ← prev
	altfel
		nodC ← NIL
		nodAnt ← prev
	sf_dacă
sf_funcție
```

```
Complexitate: O(n)


subalg cauta(mdo, c, l):

	cautNod (mdo, c, nodC, nodAnt)
	dacă nodC = NIL     atunci
		crează(l)
	altfel
		l ← [nodC].e.l
	sf_dacă
sf_subalg
Complexitate: O(n)


subalg adaugă(mdo, c, v)
	cautNod(mdo, c, nodC, nodAnt)
	dacă nodC = NIL atunci
		adaugăCheieNouă(mdo, c, v, nodAnt)
	altfel
		dacă caută([nodC].e.l, v) = fals atunci
			adaugaSfârșit ([nodC].e.l, v)
		sf_dacă
	sf_dacă
sf_subalg
Complexitate: O(mdo)


subalg adaugăCheieNouă (mdo, c, v, nodAnt)

	alocă(q)
	[q].e.c ← c
	creează ([q].e.l)
	adaugăSfârșit([q].e.l, v)
	dacă nodAnt = NIL atunci
		[q].urm ← mdo.prim
		mdo.prim ← q
	altfel
		[q].urm ← [nodAnt].urm
		[nodAnt].urm ← q
	sf_dacă
sf_subalg
Complexitate: 0(1) (adaugăSfârșit este apelat pentru o listă vidă)
```

```
subalg sterge(mdo, c, v):
	cautNod(mdo, c, nodC, nodAnt)
	dacă nodC ≠ NIL atunci
		pos ← pozitie([nodC].e.l, v)
		dacă pos ≠ -1 atunci
			sterge([nodC].e.l, pos, e)
		sf_dacă
		dacă vidă ([nodC].e.l) atunci
			stergCheie(mdo, c, nodAnt)
		sf_dacă
	sf_dacă
sf_subalg
Complexitate: O(mdo)


subalg stergCheie(mdo, c, nodAnt):
	daca nodAnt = NIL atunci
		p ← mdo.prim
		mdo.prim ← [mdo.prim].urm
		distruge([p].e.l)
		dealocă(p)
	altfel
		q ← [nodAnt].urm
		[nodAnt].urm ← [[nodAnt].urm].urm
		distruge([q].e.l)
		dealoca(q)
	sf_dacă
sf_subalg
Complexitate: 0(1) (lista de valori e vidă)
```