**Nume: Popescu Mihai**

**Grupa: 216**

**Documentatie Structuri de Date si Algoritmi**

**Cerinta proiect:**

3. TAD MultiDictionar (MultiMap) – implementare folosind o tabela de dispersie / rezolvare coliziuni prin adresare deschisa.

**1.Enuntul problemei**

Dictionar de omonime. Sa se implementeze o aplicatie care permite utilizatorului sa caute diferite explicatii ale unui cuvant intr-un dictionar si sa imbunatateasca dictionarul, utilizatorul avand dreptul de a adauga explicatii si cuvinte noi.

MultiDictionarul este folositor deoarece un cuvant (cheie) poate avea mai multe explicatii (valori).

Tabela de dispersie este de asemenea eficienta in reprezentarea MultiDictionarelor.

**2.TAD - specificare si interfata**

**a)TAD- MultiDictionar**

**domeniu**

D={d | d este un dicţionar cu elemente e = (c, v), c de tip Tcheie, v de tip TValoare}

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

**creează(d)**

pre: true

post:d∈D, d este dicţionarul vid (fără elemente)

**adaugă(d, c, v)**

pre: d∈D, c∈TCheie, v∈TValoare,

post: d’∈D, d’=d+(c, v) (se adaugă în dicţionar perechea (c, v))

**caută(d, c)**

pre: c∈TCheie

post: caută= adevărat dacă (c)∈d, caz în care v∈TValoare e valoarea asociată cheii c

fals în caz contrar, caz în care v= TValoare 0

**şterge(d, c, v)**

pre: d∈D, c∈TCheie

post: v∈TValoare

perechea (c, v) este ştearsă din dicţionar, dacă c∈ d

v= TValoare 0 în caz contrar

**dimensiune(d)**

pre: d∈D

post: dimensiune= dimensiunea dicţionarului d (numărul de elemente) ∈N

**vid(d)**

pre: d∈D

post: vid= adevărat în cazul în care d e dicţionarul vid

fals în caz contrar

**chei(d, m)**

pre: d∈D

post: m∈M, m este mulţimea cheilor din dicţionarul d

**valori(d, c)**

pre: d∈D

post: c∈Col, c este colecţia valorilor din dicţionarul d

**perechi(d, m)**

pre: d∈D

post: m∈M, m este mulţimea perechilor (cheie, valoare) din dicţionarul d

**iterator(d, i)**

{se creează un iterator pe dicţionarul d}

pre: d∈ D

post:i∈ I, i este iterator pe dicţionarul d

**distruge(d)**

pre: d∈ D

post: dicţionarul d a fost 'distrus' (spaţiul de memorie alocat a fost

eliberat)

**b)TAD Iterator**

**domeniu**

I = {i | i este un iterator pe un container având elemente de tip T Element}

**operatii (interfata TAD-ului Iterator)**

**creeaza(i, c)**

pre : c este un container

post : i ∈ I, s-a creat iteratorul i pe containerul c, (elementul

curent din iterator refera ‘primul’ element din container)

**element(i, e)**

pre : i ∈ I, curent este valid (refera un element din container)

post : e ∈ T Element, e este elementul curent din iteratie

(elementul din container referit de curent)

**valid(i)**

pre : i ∈ I

post : valid =adev, daca curent refera o pozitie valida din container

fals, contrar

**urmator(i)**

pre : i ∈ I, curent este valid

post : curent refera ‘urmatorul’ element din container fata de cel referit de curent

**3.Alegerea reprezentarii, operatiile, complexitati**

**a)Alegerea reprezentarii**

TElement:

chei:string

valori:string

MultiDictionar:

dim:intreg

numarDispersie:intreg

tabela:↑TElement

c:TCheie[0..m-1] // presupunem ca e intreg

v:TValoare[0..m-1]

+dispersie:TFunctie

( dispersie(d,c) = c mod d.numarDispersie )

Iteraror:

dict:multiDictionar

iterator:intreg

**b)Operatii si complexitati**

operatii iterator:

subalgoritm **creeaza**(i,iter): ϴ(1)

iter <- i.iterator

i.dict <- creeazaMultiDictionar(m)

sf subalgoritm

subalgoritm **distruge**(i): ϴ(1)

i.dict <- distrugeMultiDictionar()

i.iterator <- 0

sf subalgoritm

subalgoritm **valid(i)**: ϴ(1)

valid <- (i.iterator <= i.dict.dim)

sf subalgoritm

subalgoritm **urmator(i):**

cat timp i.dict.tabela.chei[i.iterator] != 0 executa ϴ(dim)

i.iterator <- i.iterator+1

sf cat timp

sf subalgoritm

subalgoritm **element(i,el):** ϴ(1)

el <- creeazaTElement(i.dict.tabela.chei[i.iterator], i.dict.tabela.valori[i.iterator])

sf subalgoritm

operatii MultiDictionar:

subalgoritm **creeaza(d,disp):** ϴ(dim)

d.numarDispersie <- disp

d.dim <- DIM

aloca(d.tabela)

sf subalgoritm

subalgoritm **distruge(d):** ϴ(dim)

dealocare(d.tabela)

d.dim <- 0

d.numarDispersie <- 0

sf subalgoritm

subalgoritm **iterator(d,i)** ϴ(1)

i<- creeazaMultiDictionarIterator()

sf subalgoritm

subalgoritm **dispersie(d,el):** ϴ(1)

dispersie <- el.hashCode() % d.numarDispersie

sf subalgoritm

subalgoritm **dimensiune(d):** ϴ(dim)

c <- 0

pentru j<-0,d.dim executa

daca d.tabela[j].chei != NIL atunci

c <- c+1

sf daca

sf pentru

dimensiune <- c

sf subalgoritm

subalgoritm **chei(d,m):** ϴ(dim)

poz <- 0

pentru j<-0,d.dim executa

daca d.tabela[j].chei != NIL atunci

m[poz + 1] <- d.tabela[j].chei

poz <- poz + 1

sf daca

sf pentru

sg subalgoritm

subalgoritm **valori(d,m):** ϴ(dim)

poz <- 0

pentru j<-0,d.dim executa

daca d.tabela[j].chei != NIL atunci

m[poz + 1] <- d.tabela[j].valori

poz <- poz + 1

sf daca

sf pentru

sg subalgoritm

subalgoritm **perechi(d,aux):** ϴ(dim)

d.chei(lista\_chei)

d.valori(lista\_valori)

pentru j=0,d.dim executa

aux.tabela[j].chei <- lista\_chei[j]

aux.tabela[j].valori <- lista\_valori[j]

sf pentru

sf subalgoritm

subalgoritm **vid(d):** ϴ(dim)

vid <-( d.dimensiune !=0)

sf subalgoritm

subalgoritm **adauga(d,cheie,valoare):** C.D : ϴ(dim)

e <- ↑TElement(cheie,valoare) C.F. : ϴ(1)

gasit <- 0 C.M. : ϴ(1)

executa

daca d.tabela[d.dispersie(e)].chei = NIL executa

d.tabela[d.dispersie(e)].chei <- e.chei

d.tabela[d.dispersie(e)].valori <-e.valori

gasit <- 1

altfel

d.dispersie(e) <- d.dispersie(e) +1

sf daca

cat timp gasit = 0

sf subalgoritm

subalgoritm **sterge(d,cheie):** ϴ(1)

//ca decizie de implementare, stergerea se va face prin inlocuirea valorilor cu NIL

e <- ↑TElement(cheie,valoare)

start <- d.dispersie(e)

pentru j=start,d.dim executa

daca d.tabela[j].chei =cheie atunci

d.tabela[j].chei <- NIL

d.tabela[j].valori <- NIL

sf daca

sf pentru

sf subalgoritm

subalgoritm **cauta(d,cheie):**

e <- ↑TElement(cheie,NIL)

start <- d.dispersie(e)

gasit <- 0

cat timp gasit = 0 executa

daca start > d.dim atunci

exit

sf daca

daca d.tabela[start].chei = cheie atunci

gasit <-1

altfel

start <- start+1

sf cat timp

cauta <- gasit

sf subalgoritm

**4.Proiectarea aplicatiei, complexitati, diagrama de apeluri**

**a)Proiectarea aplicatiei si complexitati**

In modulul TElement.h

**clasa TElement**

subalgoritm **hashCode(e):** ϴ(1)

hashCode <- (intreg)e.chei[0]

sf subalgoritm

In modulul MultiDictionar.h

**clasa MultiDictionar**

**clasa MultiDictionarIterator**

In modulul SDA.cpp

**clasa App**

subalgoritm **creaza(a)** ϴ(1)

dict<-creazaMultiDictionar

i<-creazaMultiDictionarIterator

sf subalgoritm

subalgoritm **distruge(a):** ϴ(1)

dict<-distrugeMultiDictionar

i<-distrugeMultiDictionarIterator

sf subalgoritm

subalgoritm **validate(a,cuv,def)** ϴ(1)

daca cuv.size() = 0 atunci

err <- err + “mesaj”

sf daca

daca def.size()=0 atunci

err <- err + “mesaj”

sf daca

validate <- err

sf subalgoritm

subalgoritm **run(a)**

//afisare mesaj

cat timp 1 != 0 execute

//afisare mesaj

citeste(com)

//implementeaza functionalitatile in functie de comanda citita

sf cat timp

sf subalgoritm

**subalgoritm main()**

console <- creazaApp()

consola.run()

sf subalgoritm

App.distruge()

**b)Diagrama de apeluri**

MD.distruge(md)

MDI.distruge(i)

App.validate()

TE:hashCode(e)

MD.adauga(cuv,def)

TE.creaza(e)

MD.dispersie(e)

TE.creaza(e)

MD.creaza(md)

MD.cauta(cuv)

App.creaza(a)

MDI.creaza(i)