**Documentatie**

TAD Dictionar cu chei multiple (MultiMap) – implementare folosind o tabela de dispersie / rezolvare coliziuni prin liste intrepatrunse.

*Nume: Mihok*

*Prenume: Cristian*

*Grupa: 214*

***Specificare si interfata TAD***

***Multidictionar = {md | md*** *multidictionar cu elemente de tip TElement****}***

***TElement =*** *cheie x valoare;*

*creeaza(md)*

*pre: -*

*post: s-a creat multidictionarul vid;*

*adauga(md, e)*

*pre: md: Multidictionar, e: TElement;*

*post: s-a adaugat elemental e in md;*

*sterge(md, e)*

*pre: md: Multidictionar, e: TElement;*

*post: s-a sters elemental e din md;*

*fctDispersie(md, k)*

*pre: k: interg;*

*post:*

*actualizeazaPrimLiber()*

*pre: -*

*post: actualizeazaPrimLiber=prima locatie libera;*

*iterator(md, i)*

*pre: md: Multidictionar;*

*post: se creeaza iteratorul pe md;*

***IteratorMultidictionar = {i | i*** *iterator pe* *multidictionar cu elemente de tip TElement****}***

*creeaza(i, md)*

*pre: md: Multidictionar;*

*post: s-a creat iteratorul* ***i*** *cu vid iar current refera primul element din multidictionar;*

*getCurrent(i, e)*

*pre: i: IteratorMultidictionar*

*post: e:TElement este elemental current din iteratie;*

*vida(i)*

*pre: i: IteratorMultidictionar*

*post: valid=true daca current refera o pozitie valida din Multidictionar;*

*valid=false in caz contrar;*

*getNext(i)*

*pre: i: IteratorMultidictionar;*

*post: current refera urmatorul element din colectie;*

*getKey(i)*

*pre: i: IteratorMultidictionar;*

*post: returneaza cheia elementului current din iteratie;*

*getValue(i)*

*pre: i: IteratorMultidictionar;*

*post: returneaza valoarea elementului current din iteratie;*

***Implementare TAD***

**MultiDictionar**

*Subalgoritm creeaza(md)*

*dim<-MAX*

*primLiber<-0*

*pentru i<- 0, md.dim exec*

*md.keys[i]<- -1*

*md.next[i]<- -1*

*md.values[i]<- “”*

*sf\_pentru*

*sf\_creeaza*

*subalgoritm actualizeazaPrimLiber(md)*

*cattimp (md.primLiber <= md.dim si md.keys[primLiber <> -1])*

*md.primLiber<-md.primLiber + 1*

*sf\_cattimp*

*sf\_actualizeazaPrimLiber*

*functie adauga(md, key, value)*

*dis<-md.fctDispersie(key)*

*daca (md.keys[dis]=-1) at*

*md.keys[dis] = key*

*md.values[dis] = values*

*daca (md.primLiber = dis) at*

*actualizeazarimLiber()*

*adauga<-true*

*altfel*

*cattimp (dis <> -1 si (md.keys[dis] <> key sau md.values[dis] <> values))*

*prev <- dis*

*dis = md.next[dis]*

*sf\_cattimp*

*daca (dis <> -1)*

*adauga <- false*

*altfel*

*daca (md.primLiber = md.dim)*

*adauga<-false*

*altfel*

*md.keys[md.primLiber]=key*

*md.values[md.primLiber]=values*

*md.next[prev]=md.primLiber*

*md.actualizeazaPrimLiber()*

*adauga<-true*

*sf\_daca*

*sf\_daca*

*sf\_adauga*

*functie sterge(md, key, value)*

*prev<- -1*

*dis<- md.fctDispersie(key)*

*cattimp (dis <> -1 si (md.keys[dis] <> key sau md.values[dis] <> values))*

*prev<-dis*

*dis<- this.next[dis]*

*sf\_cattimp*

*daca(dis=-1)*

*sterge<-false*

*altfel*

*done<-false*

*repeta*

*pprev<-dis*

*p<-md.next[dis]*

*cattimp(p<> -1 si md.fctDispersie(md.keys[p])<>dis)*

*pprev<-p*

*p<-md.next[p]*

*sf\_cattimp*

*daca(p=-1)*

*done<-true*

*altfel*

*md.keys[dis]<-md.keys[p]*

*md.values[dis]<-md.values[p]*

*prev<-pprev*

*dis<-p*

*sf\_daca*

*pana\_cand (done <> true)*

*daca(prev <>-1)*

*md.next[prev]<- md.next[dis]*

*sf\_daca*

*md.keys[dis]<- -1*

*md.next[dis]<- -1*

*mad.values[dis]<- “”*

*daca(dis < md.primLiber)*

*md.primLiber<-dis*

*sterge<-true*

*sf\_daca*

*sf\_sterge*

**MultiDictionarIt**

*subalgoritm creeaza(i, mdict)*

*i.current <- 0*

*md<- mdict*

*sf\_creeaza*

*subalgoritm prim(i)*

*current <- 0*

*cattimp(md.keys[current]=-1)*

*current<-current +1*

*sf\_cattimp*

*subalgoritm urmator(i)*

*current <-current+1*

*cattimp(md.keys[current]=-1 si current<=MAX)*

*current <-current+1*

*sf\_cattimp*

*sf\_urmator*

*functie valid(i)*

*daca(currnet < MAX si md.keys[current]<>-1)*

*valid<- true*

*altfel*

*valid<-false*

*sf\_daca*

*functie getCurrent(i)*

*getCurrent<-current*

*sf\_getCurrent*

*functie getKey(i, it)*

*getKey<-md.keys[it*

*sf\_getKey]*

***Aplicatie***

*Creati o aplicatie care sa gestioneze hainele din garderoba unui restaurant. Multimea de bunuri ale unei persoane se depoziteaza pe o singura cheie.*