Asszociatív tömb típus

Asszociatív tömb:

Típus értékek:

Map azon asszociatív tömbök halmaza, amely tömböknek az elemei $\mathbb{Z} \times \mathbb{S}$ típusú párok

Típus műveletek:

m.setEmpty() m : Map
//kiüríti az asszociatív tömböt

c := m.count() m : Map, c : $\mathbb{N}$
//megadja az elemek számát

m.insert(e) m : Map, e : $\mathbb{Z} \times \mathbb{S}$
//új elemet tesz be, ha a kulcsa még nem létezik

m.erase(key) map : Map, key : $\mathbb{Z}$
// törli az adott kulcsú elemet, ha a kulcs létezik, különben hiba

l :=m.in(key) m : Map, key : $\mathbb{Z}$, l : $\mathbb{L}$
//lekérdezi, van-e adott kulcsú elem

data := m.operator[](key) m: Map, key : $\mathbb{Z}$, data : $\mathbb{S}$
// lekérdezi az adott kulcsú elem adatát, ha a kulcs létezik, különben hiba

Típus reprezentáció:

vec: Item[] – az elemeket kulcsuk szerint rendezetten tároló tömb,

ahol Item = rec(key: $\mathbb{Z}$, data: $\mathbb{S}$)

Típusműveletek implementációja:

m.setEmpty()

vec := <>

c := m.count()

c := |vec|

m.insert(e) l : $\mathbb{L}$, ind : $\mathbb{N}$

l, ind := logSearch(vec, e.key)

¬l

vec := vec[1 .. ind-1] ⊕ e ⊕ vec[ind+1 .. |vec|]) | —

m.erase(key) l : $\mathbb{L}$, ind : $\mathbb{N}$

l, ind := logSearch(vec, key)

l

vec := vec[1 .. ind-1] ⊕ vec[ind+1 .. |vec|]) | **Hiba**: nem létező kulcs

l := m.in(key)

l, ind := logSearch(vec, key) | ind : $\mathbb{N}$

data := m.operator[](key) //m[key] alakú hivatkozás

l, ind := logSearch(vec, key)

l

data := vec[ind].data | **Hiba**: nem létező kulcs

Asszociatív tömb típus

Több művelet egy kulcs szerinti kereséssel indul, amelyhez a logaritmikus keresés algoritmusát használjuk. Ezt még egy lépéssel ki is egészítjük: ha nincs benne a keresett kulcs a tömbben, akkor az első nála nagyobb elem sorszámát adjuk vissza. Ez az utolsó elem utáni sorszám lesz, akkor ha nincs a keresett kulcsnál nagyobb a tömbben.

A = (vec : Item*, key : $\mathbb{Z}$, l : $\mathbb{L}$, ind : $\mathbb{N}$)

Ef = (vec=vec_0 $\wedge$ key=key_0 $\wedge$ $\forall i \in [1 .. |vec|-1]$: vec[i].key < vec[i+1].key)

Uf=(Ef $\wedge$ l = $\exists i \in [1 .. |vec|]$: vec[i].key = key $\wedge$
(l $\rightarrow$ ind$\in$[1 .. |vec|] $\wedge$ vec[ind].key = key) $\wedge$
($\neg$l $\rightarrow$ $\forall i \in$ [1..ind-1]: vec[i].key < key $\wedge$ $\forall i \in$[ind.. |vec|] : vec[i].key > key))

l, ind := logSearch(vec, key)