OBJEKTUM ELVŰ ALKALMAZÁSOK FEJLESZTÉSE
Dokumentáció a 2. házi feladathoz

Név: Soós Bálint
Neptun kód: HDX9MU
Elérhetőség: soba95@inf.elte.hu

Csoport: 8.
Feladatszám: 7.
2015. október 15.

Feladatleírás

Készítsen egy halmaz típust! A halmazt rendezett láncolt listával ábrázolja! Implementálja a szokásos műveleteket (elem betétele, kivétele, benne van-e egy adott elem, üres-e), egészítse ki az osztályt a halmaz tartalmát kiíró operátor«-ral! Definiáljon olyan barát-operátorokat is, amely kiszámítja két halmaz szimmetrikus differenciáját és metszetét! A metszet műveletigénye: O(m+n), ahol m és n a két halmaz elemszáma.

Halmaz típus

A feladat megoldásához definiálni kell egy egész számokat tartalmazó halmaz típust.

Típusérték-halmaz

Egész számok halmaza. (beleértve az üres halmazt is)

Típus-műveletek

1. *Elem berakása a halmazba*

   A felhasználó által megadott egész számot berakjuk a halmaz elemei közé, így a halmaz mérete eggyel nő. Érdemes megjegyezni, hogy a műveletet csak akkor kell elvégezni, ha az adott elem még nem szerepel a halmazban.

2. *Elem kivétele a halmazból*

   Először megvizsgáljuk, hogy szerepel-e az adott elem a halmazban. Ha igen, akkor az elemet eltávolítjuk a halmazból, így mérete eggyel csökken.

3. *Üres-e a halmaz*

   Megvizsgáljuk, hogy a halmazban van-e elem, azaz a halmaz üres, ha mérete 0.

4. *Elem benne van-e a halmazban*

   Szerepel-e az adott elem a halmaz elemei között.

5. *Halmaz kiírása*

   Megjelenítjük a halmaz méretét, és az elemek listáját.

6. *Két halmaz metszete*

   Két halmaz metszete egy olyan halmaz, melynek elemei mindkét halmaznak is elemei.

7. *Két halmaz szimmetrikus differenciája*

   Két halmaz szimmetrikus differenciája a halmazok úniójának és metszetének különbsége.

Reprezentáció

Az egész számokat tartalmazó halmazt egy egyirányú, fejelemes, láncolt listával ábrázoltam. Az halmazműveletek hatékonyságát szem előtt tartva a lista elemei rendezve vannak. A rendezéshez a beszúró rendezés algoritmusát használtam.

Implementáció

1. *Elem berakása a halmazba (put)*

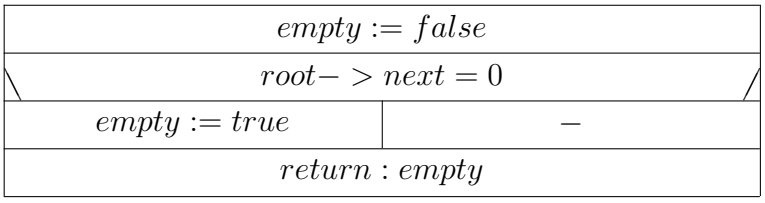
   A listán végighaladva a beszúró algoritmust alkalmazva a keresett pozícióban létrehozok egy új elemet és az előtte lévő elem next-jét az új elemre állítom, az új elem nextjét pedig az utána következő elemre.

2. *Elem kivétele a halmazból (remove)*

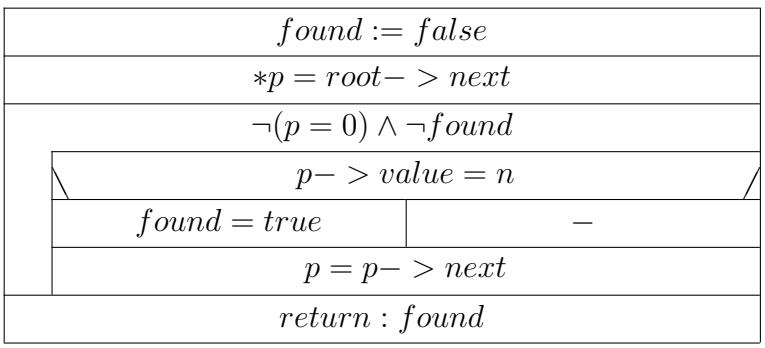
   Egy elem kivételénél ugyanazt az algoritmust használom, mint az elem berakásánál, csak itt a megadott pozícióban kiveszem az elemet, az előtte álló elem mutatóját a következő elemre állítom, illetve az adott elemet törlöm.

3. *Üres-e a halmaz (isEmpty)*

   Ha a lista fejelemét követő első elem nullpointer, akkor a halmaz üres.



4. *Elem benne van-e a halmazban (isContain)*



5. *Halmaz kiírása (print)*

   Lista elemein végiglépdelve kiírjuk a kimenetre az elemek értékeit.

6. *Két halmaz metszete (intersection)*

   Az A és B halmaz metszetét (felhasználva, hogy a listák rendezettek és lehetnek üresek is) úgy kapjuk meg, hogy a két listán párhumazosan lépdelve, amíg valamelyik el nem fogy:

   1) ha A halmaz eleme nagyobb, mint a B halmaz eleme, akkor B halmaz következő elemére lépek
   2) ha A halmaz eleme kisebb, mint a B halmaz eleme, akkor A halmaz következő elemére lépek
   3) ha A halmaz eleme egyenlő B halmaz elemével, akkor metszetelemet találtam

7. *Két halmaz szimmetrikus differenciája (symDef)*

   Az A és B halmaz szimmetrikus differenciáját (felhasználva, hogy a listák rendezettek és lehetnek üresek is) úgy kapjuk meg, hogy a két listán párhumazosan lépdelve, amíg valamelyik el nem fogy:

   1) ha A halmaz eleme nagyobb, mint a B halmaz eleme, akkor B halmaz elemét kiírom, majd a következő elemére lépek
   2) ha A halmaz eleme kisebb, mint a B halmaz eleme, akkor A halmaz elemét kiírom, majd a következő elemére lépek
   3) ha A halmaz eleme egyenlő B halmaz elemével, akkor metszetelemet találtam, mindkettővel a következő elemükre lépek

   Ha elfogy valamelyik halmaz, akkor a másik halmazban lévő maradék elemek a szimmetrikus differencia elemei, tehát azokat is kiírom.

Tesztelési terv:

I) A feladat specifikációjára épülő (fekete doboz) tesztesetek:

   1) 1,2,3 hosszúságú halmazok létrehozása a put metódussal
   2) 1,2,3 hosszúságú halmazok esetén a remove metódus kipróbálása
   3) 0 hosszúságú halmazok esetén a remove metódus hívása
   4) 0,1,2 hosszúságú halmazok esetén az isEmpty és isContain metódus kipróbálása
   5) 0,1,2 hosszúságú halmazok esetén az isEmpty és isContain metódus kipróbálása
   6) 0,1,2 hosszúságú halmazok esetén a print metódus kipróbálása
   7) két azonos (nem 0) hosszúságú halmazok metszete
   8) két különböző (nem 0) hosszúságú halmazok metszete
   9) két különböző (az egyik 0) hosszúságú halmazok metszete
   10) két 0 hosszúságú halmazok metszete

II) A megoldó programra épülő (fehér doboz) tesztesetek:

   1) Halmazok létrehozása (max 10 db halmaz)
   2) Már meglévő 10 db halmaz esetén új halmaz létrehozás (max 10 db halmaz lehet)
   3) Halmazok közti váltás, ha valid, illetve ha invalid halmaz számot adunk meg

4) Új elem berakása a halmazba, ha az még nem létezik.
5) Új elem berakása a halmazba, ha az már létezik.
6) Elem törlése a halmazból, ha az létezik.
7) Elem törlése a halmazból, ha az nem létezik.

Osztály

Az egész számokat tartalmazó halmaz típust egy Set nevű osztály segítségével valósítottam meg.

Set	**Exceptions**
-Node : struct -root : Node	
+Set() +Set(int) +Set(Set&) +operator=(Set&) : Set +operator+(Set&) : Set +operator-(Set&) : Set +operator«(Set&) : Set +put(int) : void +remove(int) : void +print() : void +intersection(Set&) : void +symDef(Set&) : void +isContain(int) : bool +isEmpty() : bool +getSize() : int +getItems() : int*	+EMPTY +CONTAIN +INVALID_ITEM

Az osztály-definíciót a set.h fejállományban helyeztem el.