

# OBJEKTUM ELVŰ ALKALMAZÁSOK FEJLESZTÉSE

## Dokumentáció az 1. házi feladathoz

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

**Csoport:** 8.  
**Feladatszám:** 7.  
2015. október 2.

### Feladateleírás

Valósítsa meg az egész számokat tartalmazó halmaz típust! A halmazt dinamikusan lefoglalt tömb segítségével ábrázolja! Implementálja a szokásos műveleteket (elem be-  
tevése, kivétele, üres-e a halmaz, egy elem benne van-e a halmazban), valamint két  
halmaz metszetét, továbbá egy halmaz kiírását, és végül a másoló konstruktort és az  
értékadás operátort! Törekedjen a metszetképzés műveletigényének minimalizálására,  
a dokumentációban mutasson rá a saját megoldásának műveletigényére!

### Típusérték-halmaz

Feladatom egy egész számokat tartalmazó halmaz típus megvalósítása, azaz olyan  
(esetünkben dinamikus) tömb létrehozása, amelyben az elemek nem ismétlődhetnek.

### 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 ele-  
mei.

## Reprezentáció

Az egész számokat tartalmazó halmazt egy dinamikusan lefoglalt, egydimenziós, 1-től  $n$ -ig indexelt tömbbel ábrázoltam, amelynek méretét (tömb hosszúságát) egy külön változóban tárolom. Az halmazműveletek hatékonyságát szem előtt tartva a tömb elemei rendezve vannak. A rendezéshez a beszűrő rendezés algoritmusát használtam.

## Implementáció

### 1. Elem berakása a halmazba (*put*)

A tömb hosszúságát eggyel megnövelve (dinamikusan lefoglalt tömb esetén ehhez először törölni kell a tömböt) a beszűrő rendezés algoritmusát felhasználva berakom a tömbbe a megadott elemet.

Ehhez létrehozok egy ideiglenes, az eredeti tömb hosszúságánál eggyel hosszabb statikus tömböt, amelybe a beszűrő algoritmus során bepakolom az elemeket, majd az algoritmus végén átmásolom az eredeti tömbbe.

### 2. Elem kivétele a halmazból (*remove*)

Egy elem kivételére létrehozok egy statikus, az eredeti tömb hosszúságánál eggyel rövidebb ideiglenes tömböt, amibe az adott elem kivételével átmásolom a tömb tartalmát.

Az algoritmus végeztével az ideiglenes tömb tartalmát átmásolom az eredeti tömbbe.

### 3. Üres-e a halmaz (*isEmpty*)

Ha a tömb hosszúsága 0, akkor a tömb üres.

$empty := false$	
$size = 0$	
$empty := true$	—
$return : empty$	

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

$items[]$ : halmaz elemeit tartalmazó dinamikusan lefoglalt tömb,  $size$ : a tömb hosszúsága (halmaz mérete)

$found := false$	
$i := 0$	
$i < size \wedge \neg found$	
$items[i] = n$	
$found = true$	—
$return : found$	

5. *Halmaz kiírása (print)*

Tömb elemein végiglépdelve kiírjuk a kimenetre az elemek értékeit. (a programban a tömb hosszúságát is kiírjuk)

$i := 0$
$i < size$
$output : items[i]$
$i := i + 1$

6. *Két halmaz metszete (intersection)*

Az s és h halmaz metszete (felhasználva, hogy a tömbök rendezettek és lehetnek üresek is).

$\neg s.isEmpty() \wedge \neg h.isEmpty()$	
$iOfS := s.size - 1$	—
$iOfH := h.size - 1$	
$iOfS \geq 0 \wedge iOfH \geq 0$	
$s.items[iOfS] > h.items[iOfH]$	
$iOfS := iOfS - 1$	
$s.items[iOfS] < h.items[iOfH]$	
$iOfH := iOfH - 1$	
$s.items[iOfS] = h.items[iOfH]$	
$output : s.items[iOfS]$	
$iOfS := iOfS - 1$	
$iOfH := iOfH - 1$	

Metszetképzés műveletigénye:

c1: függvény kiértékelés

c2: értékadás, kivonás

c3: reláció kiértékelés

c4: tömb elemének kiolvasása

c5: output

n1, n2: tömbök hossza

M: műveletigény

minimális M (ha legalább egy tömb üres):  $M = 2 c1$

maximális M: Ha nincs metszetelem:  $M = 2 c1 + 4 c2 + (n1+n2)*(c3 + 2 c4 + 2 c1)$

M:  $O(n1+n2)$

## Osztály

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

Set	Exceptions
-items: int* -size: int	
+Set() +Set(int) +Set(Set&) +operator=(Set&) : Set +put(int) : void +remove(int) : void +print() : void +intersection(Set&) : void +isContain(int) : bool +isEmpty() : bool +getSize() : int +getItems() : int*	+EMPTY +CONTAIN +INVALID_ITEM

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

### 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.