HALMAZOK

A modern programozási nyelvekben a halmazok olyan objektumok, melyek tetszőleges – elemi és összetett típusú – elemeket tartalmazhatnak. Az elemszámnak csak a program rendelkezésére álló memória szab határt.

Könyvünkben a halmazokat elsősorban halmazfelsorolás készítéséhez használjuk (lásd alább). Nem alkalmazunk

- halmazműveleteket (unió, metszet, részhalmaz vizsgálata stb.),
- összetett típusú adatokat (például struktúrákat) tartalmazó halmazokat,
- olyan összetett adatszerkezeteket (struktúrák, tömbök stb.), melyek elemei halmazok.

A felsorolásban szereplő eseteket a *Kollekciók alkalmazása a feladatmegoldásokban* című példatár mutatja be. Javasoljuk az Olvasónak, hogy mihamarabb ismerkedjen meg a kollekciókkal, mert lényegesen leegyszerűsítik a feladatok megoldását. A példatár kitér a gráfok halmazok segítségével történő tárolására is.

Egyes programozási nyelvek támogatják a rendezett halmaz adatszerkezet használatát. Alkalmazása esetenként nagymértékben megkönnyíti a feladatok megoldását (lásd például az *Alma* feladatot).

Létrehozásakor a halmazt inicializálhatjuk az elemek felsorolásával, illetve egy másik halmaz, tömb vagy egyéb kollekció megadásával:

Halmaz = Új Halmaz(Elemtípus: *típus*) {*az elemek felsorolása*}
Halmaz = Új Halmaz(Elemtípus: *típus*) (*egy kollekció azonosítója*)

Ha az inicializáló sorozat egyforma elemeket is tartalmaz, akkor azok csak egy példányban kerülnek be az új halmazba.

A halmazobjektumok legfontosabb metódusai közé tartozik az elemek törlése (az eredmény üres halmaz), egy elem hozzáadása, illetve kivétele. Lekérdezhetjük továbbá az elemek számát, illetve hogy a halmaz tartalmaz-e egy megadott elemet, illetve üres halmaz-e. Ezek a műveletek általában nagyon gyorsak.

> Visual Basic-ben a halmaz inicializálásánál, az elemek felsorolása előtt írjuk ki a *From* kulcsszót! Például:
> `Dim Halmaz As New HashSet(Of Integer) From {1, 2, 3}`
> Az *Add* és a *Remove* metódusok olyan függvények, melyek visszatérési értéke jelzi a művelet végrehajtásának eredményét (már létező elem esetén az *Add* értéke *False*, a *Remove* pedig *True*).
> Rendezetlen halmaznál a *Contains* és a *Remove* metódus futásideje O(1) nagyságrendű (!), csakúgy, mint az *Add* metódusé, ha ez utóbbi nem igényli a kapacitás növelését. A kapacitás növelésekor az *Add* művelethez O(*n*) idő szükséges. A rendezett halmaz *Contains* és *Remove* metódusainak futásideje: O(log *n*).

> A C++11 nyelvben az elemeket rendezetten tároló *set* és az elemek rendezettségét nem megőrző *unordered_set* halmaztípusok közül választhatunk. Az első esetben a hozzáadás (*insert*), keresés (*find*) és törlés (*erase*) műveletek időszükséglete O(log *n*). Ezzel szemben a második esetben O(1) a szükséges idő, ha az elem még nem szerepel a konténerben, és O(*n*), ha már benne van. Az *insert()* tagfüggvény visszatérési értéke (*pair<iterator, bool>*) a C++-ban is jelzi, hogy megtörtént-e az elem beillesztése a halmazba, továbbá tartalmaz az elemre hivatkozó iterátort.
> A halmazokat a létrehozásuk során a szokásos módon, az inicializációs lista megadásával láthatjuk el elemekkel: `set<int> halmaz {1, 2, 3};`
> A halmazok *find()* tagfüggvénye által visszaadott iterátor értékéből következtethetünk arra, hogy egy adat benne van-e halmazban vagy sem:
> `if (halmaz.find(elem)!=end(halmaz)) // tartalmazza!`

A halmazok elemeire gyakran indexükkel hivatkozhatunk, bár a programozási nyelvek nem garantálják, hogy az indexelés megfelel a hozzáadás sorrendjének. A halmaz egy elemének értéke természetesen nem módosítható[102] (legfeljebb törölhető).

Az elemeket a számlálós ciklus mellett iterátoros ciklussal is lekérdezhetjük:

```
CIKLUS MINDEN Elem-re a Halmaz-ban
   ...
CIKLUS VÉGE
```

Visual Basic-ben a halmazelemek indexelése 0-val kezdődik.

C++-ban a halmazok nem támogatják az indexelés operátorát. Egy halmaz bejárásához iterátorra épülő számlálós ciklust vagy tartományalapú *for* ciklust használhatunk:

```
set <int> s;
...
for (auto p=begin(s); p!=end(s); p++)
   cout << *p << " ";
...
for (int e : s)
   cout << e << " ";
```

Amennyiben egy algoritmusban indexekkel kell dolgoznunk, a halmaz elemeivel egy vektor konténert is inicializálhatunk: `vector<int> v (begin(s), end(s));`

A **multihalmaz** olyan halmaz, amely egy elemet többször is tartalmazhat. A multihalmaznál az elem értéke mellett feljegyezzük a gyakoriságát, melyet az elem *multiplicitásának* nevezünk. A multihalmazokat gyakran asszociatív tömbökben[103] tároljuk. Az asszociatív tömböket a *Programozási ismeretek haladóknak* című tankönyvben ismertetjük. Az asszociatív tömbök sok versenyfeladat megoldását segítik, érdemes velük megismerkedni. Számos hozzájuk kapcsolódó feladatot tartalmaz a *Kollekciók alkalmazása a feladatmegoldásokban* című példatár.

Visual Basic-ben a multihalmazt a *Dictionary* típussal (asszociatív tömbbel) valósíthatjuk meg, melyben a halmazelemeket kulcsként (*Key*), az ismétlődések számát pedig értékként (*Value*) tároljuk.

C++-ban *multiset* és *unordered_multiset* konténerek valósítják meg a multihalmazt.

Halmazfelsorolás készítése

Halmazfelsorolásnak nevezzük az olyan sorozatot (például tömböt), melynek nincsenek egyforma elemei.[104]

Tetszőleges tömbből úgy készíthetünk halmazfelsorolást, hogy elemeit halmazban gyűjtjük össze:

```
Halmaz.Töröl()
CIKLUS I=Tömb.Kezdőindex-től Tömb.LegnagyobbIndex-ig
   Halmaz.Add(Tömb(I))
CIKLUS VÉGE
```

[102] Elemi adattípusú elemek esetén.

[103] Az asszociatív tömböt gyakran szótárnak (*Dictionary*) hívják.

[104] Lásd a *Programozási ismeretek* tankönyv 241. oldalán.