1.5 Tömbök

Tömbök

Korábban már volt szó konkrét tömbökrl (pl.: parancssori paraméterek tömbje). Az akkor tárgyaltak természetesen általánosságban is igazak a tömbökre. Egy rövid összefoglaló: A tömbök azonos típusú, de egynél több érték tárolására vannak kitalálva. Az elemek száma azonban rögzített, tehát adott számú, azonos típusú elemet tartalmazó adattípusról van szó. Az indexelés 0-val kezddik, ahogy az ismers lehet *Programozás alapjai* tárgyból.

> ***Megjegyzés**: Habár Javaban, C-ben 0-tól indexelünk, nem feltétlenül van ez így minden nyelvben. Vannak olyan nyelvek, melyek egytl kezdik az indexelést. Ezen nyelvekrl itt találsz egy összefoglaló listát.*

Fontos lehet, hogy a C-vel ellentétben itt nagyon szeretjük a tömböket, hiszen a Java (mivel menedzselt nyelvrl van szó, azaz a kód nem közvetlenül a hardveren fut, hanem a JVM-en, ahogy ezt korábban tárgyaltuk) teljes kör indexhatár-ellenrzést végez, kivétellel jelzi, ha alul- vagy túlindexelés történik.

A Java-beli tömböket többféleképpen is létre lehet hozni. Statikusan, ha elre tudjuk a tömb elemeit, vagy pedig dinamikusan, amikor nem feltétlen tudjuk a tömb elemeinek értékét. Mind primitív típusból, mind pedig osztályokból létrejöv objektumok esetén használhatunk tömböket. Tömböket dinamikusan, a `new` operátorral lehet létrehozni, mérete a már ismert `length` tulajdonsággal lekérdezhet.

Egydimenziós tömbök

Az egyetlen kiterjedéssel (dimenzióval) rendelkez tömböket szokás vektornak is nevezni, ahogy ez a kifejezés matematikából ismers lehet. A tömböt deklarálni a következképp lehet:

```
típus tömbnév[];
típus[] tömbnév;
```

A két deklaráció között semmilyen különbség sincs. Futás közben a már említett `new`operátorral tudjuk definiálni:

```
tömbnév = new típus[méret];
```

Egy konkrét példa:

```
int[] x = new int[5];
```

Ennek a tömbnek az elemei: x[0], x[1], x[2], x[3], x[4]. A tömböt legegyszerbben egy `for`ciklussal tölthetjük fel.

A Java lehetvé teszi, hogy a tömböket a definiálás során konstans értékekkel töltsük fel. Ilyenkor a tömböt a fordító hozza létre.

```
típus[] tömbnév = { konstans1, konstans2 }
// egy konkrét példa:
int[] arr = { 1, 2, 3, 4, 5 };
```

Adott tömb mérete minden esetben egy nemnegatív egész szám, melynek értéke nem lehet nagyobb, mint a lehetséges legnagyobb index, ez a JVM-tl függ, és mivel az indexelés `int`értékekkel történik, ezért az elméleti legnagyobb elemszám `Integer.MAX_VALUE` (ez egy definiált konstans), azonban ez a gyakorlatban egy ettl 5-8 értékkel kisebb szám. Derítsük ki, hogy a saját JVM-ünk esetében mekkora ez a szám: ehhez hozzunk létre egy tömböt, és próbáljuk meg lefordítani, és lefuttatni az elkészült fájlt:

```
String[] tomb = new String[Integer.MAX_VALUE - 4];
```

Többdimenziós tömbök

Analóg módon az egydimenziós tömbökhöz, illetve a korábban, Programozás alapjain tanultakhoz:

```
típus[][]...[] tömbnév;
típus tömbnév[][]...[];
```

Két dimenziós tömb létrehozására egy példa:

```
típus[][] tömb;
tömb = new típus[méret1][méret2];
```

Egy konkrét példa kétdimenziós tömb létrehozására és feltöltésére. A tömböt legegyszerbben egy `for` ciklussal tölthetjük fel.

```
int[][] tomb;
tomb = new int[10][9];

for (int i=0; i < tomb.length; i++){
    for (int j = 0; j < tomb[i].length; j++) {
        tomb[i][j] = (i+1)*(j+1)*9;
    }
}
```

A Java lehetvé teszi, hogy a tömböket a definiálás során konstans értékekkel töltsük fel. Ilyenkor a tömböt a fordító hozza létre

```
int[][] matrix = { { 1, 2 }, { 3, 4 } };
```

Tömbök másolása

Mivel a tömbök referencia típusúak (elz gyakorlat), ezért egy egyszer értékadás esetén csak a referenciát (azaz a tömb címét) másoljuk át egy másik változóba, a konkrét, memóriában létez tömböt nem. Tényleges tömbmásolást kézzel is megvalósíthatunk, egyszer for ciklus segítségével, de létezik egy JDK-ban már elre elkészített metódus épp erre a célra: a tényleges másolás a `System.arraycopy()` metódussal lehetséges. Ennek fejléce (javadoc):

```
public static void arraycopy(Object forrás, int forrásKezdetiElem, Object cél, int kezdetiPozíció, int hoss
z)
```

Egy konkrét példa tömbök másolására:

```
int tomb1[] = { 30, 31, 1, 2, 3, 4, 5, 6, 7 };
   int tomb2[] = { 29, 0, 0, 32, 33 };

   System.arraycopy(tomb1, 0, tomb2, 1, 2);

   for (int i=0; i<tomb2.length; i++){
       System.out.print(tomb2[i] + " ");
   }
   System.out.println();
```

Ennek kimenete: `29 30 31 32 33`.