Tömb (Array)

Az egyik leggyakrabban használt programozási eszköz a tömb. Amikor sok azonos típusú adatot kell tárolni, amelyek között még kapcsolat is van (valamilyen szempont alapján összetartoznak), akkor nem célszerű mindegyikhez egy-egy változót deklarálni, mert a feldolgozásukat ez nem segíti.

Rossz megoldás: a1, a2, a3, …, a100 változók használata. (*ezt gyorsan el kell felejteni*)

Jó megoldás: Egy 100 elemű tömböt használjunk az adatok tárolásához!
Az a[0], a[1], a[2], …, a[99] módon tárolt adatokkal sokkal egyszerűbben lehet dolgozni.

A tömb adatszerkezet előnye, hogy egyetlen változót kell létrehozni, amelyben több adatot tárolhatunk. Az egyes adatelemeket egy index különbözteti meg egymástól.

A indexelés minden esetben 0-tól kezdődik. Az indexet a név után szögletes zárójelek közé kell írni.

Vektor (1-dimenziós tömb)

Akkor nevezzük így a tömböt, ha csak egy indexet használunk az elemek megkülönböztetésére..

A tömb létrehozása két részletben történik. Az első egy tömb hivatkozás (referencia) megadása.

```
int[] szamok;
```

Megadtuk az azonosítóját (szamok), de még nem lehet tudni, hogy hány számot akarunk tárolni itt. Második lépésben megadjuk az elemek számát, vagy felsoroljuk a tárolandó adatelemeket.

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

Ezzel a memóriában lefoglalunk öt int típusú egész szám tárolásához szükséges területet.

A két lépést **össze is lehet vonni**, de ezt nem minden esetben érdemes.

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

Az adatok felsorolásával is létrehozhatjuk a tömböt.

```
int[] szamok = {3,3,3,4,5};
```

A kétféle módszer kombinálható egymással.

```
int[] szamok = new int[5] {3,3,3,4,5};
```

```
int[] szamok = new int[] {3,3,3,4,5};
```
is helyes.

A tömbök használatának számtalan előnye van. Az így tárolt adatok feldolgozása gyors.

Példa: Tömbben tárolt számokat összegezzünk!

```
for (int i = 0; i < szamok.Length; i++)
{
    összeg += szamok[i];
}
```

A **Length** tulajdonság megadja a tömb elemeinek számát, ezért használtuk a for ciklus fej részében.

Kiegészítés

A .Net keretrendszerben lévő Array osztály a metódusai révén alapvető problémákra kész megoldásokat nyújt.

A fenti példában lévő szamok tömb elemeinek összegét a **szamok.sum()** használatával közvetlenül megkapjuk. A sum() az Array osztály egy u.n. tagfüggvénye, amely összegzi a tömb elemeket és visszatér az összeg értékével.

Tömb bejárása az elemein keresztül

Ha csak lekérni akarjuk az elemeket, azaz nem akarjuk módosítani az értékeket, akkor „**foreach**” ciklussal is dolgozhatunk.

```
foreach (int elem in szamok)
{
    összeg += elem;
}
```

vagy

```
foreach (var elem in szamok)
{
    összeg += elem;
}
```

Az ***elem*** itt egy formális változó (a név tetszőleges lehet). Először felveszi a tömb első elemének ***értékét***, majd a következő ciklusban a másodikat, és így tovább, ameddig végig nem halad az összes elemen. Így minden tömb elemmel végrehajtódik a ciklus törzsében lévő utasítás. Jelen esetben az összegzés.

A típus helyén használhatjuk a **var** szócskát. Ilyenkor a program „kitalálja” a szükséges típust (jelen esetben ez int lesz).

Mátrix, táblázat (kétdimenziós tömb)

```
int[,] matrix = new int[2,3]; vagy
int[,] matrix = new int[2,3] = {{1, 2, 3},{4, 5, 6}}; vagy
int[,] matrix = new int[,] = {{1, 2, 3},{4, 5, 6}}; vagy
...
```

Látható, hogy az elemeket két index segítségével tudjuk azonosítani. Legegyszerűbben egy táblázatként képzelhető el (sorindex, oszlopindex), viszont a memóriában sorfolytonosan tárolódnak a mátrix elemei.

A mátrix elemeinek számát itt is a **Length** tulajdonság adja meg. Emellett szükség van az egyes dimenziók méretének lekérdezésére is a **GetLength(dimenziósorszám)** metódussal.

Megjegyzés

Készíthetünk olyan egydimenziós tömböt, amelynek elemei szintén egydimenziós tömbök. Ez egy alternatív megoldás lehet a kétdimenziós tömbre.

Kiegészítés

Az Array osztály néhány jól hasznosítható metódusa:

`Array.Clear(szamok)`	A tömbben tárolt értékek törlése. Lehet csak egy részt is törölni, ha indexeket is megadunk
`Array.Copy(szamok, szamok2)`	A tömb másolása. Itt is lehet csak egy részt átmásolni
`Array.Resize(ref szamok, N)`	A tömb méretének megváltoztatása
`Array.Reverse(szamok)`	A tömb elemei fordított sorrendben legyenek
`Array.Sort(szamok)`	A tömb elemeinek növekvő rendezése
`Array.IndexOf(szamok, 8)`	Hol van a keresett érték?