5. Tömbök - I rész

Eddig olyan esetekkel foglalkoztunk, ahol egy változónévhez egy értéket tudtunk hozzárendelni. (Skalár változó). A tömb azonos típusú, véges számú elemből épül fel. Olyan közvetlen elérésû adatstruktúra, amelynek egy elemét a tömb nevével és az utána következô index-szel tudjuk elérni - alapesetben.

Milyen jellemzôi vannak egy tömbnek C-ben ? A tömbnek öt jellemzôje van :

- *Tárolási osztály specifikátor*
- *Típus*
- *Név*
- *Méret*
- *Dimenzió*

Típus :

Az eddig megismert változó típusok valamelyike lehet.

Név :

A tömb nevének megválasztásával kapcsolatban a 2.2 pontban leírtak érvényesek.

Méret :

A tömb definíciónál a tömb típusa után a nevét adjuk meg és utána az elemek számát - "[" ill. "]" között. **C -ben a tömb indexelése szigorúan a 0 - dik elemtôl kezdôdik.** Indexelésnek nevezzük azt a mûveletet, amikor a tömb neve után "[" és "]" között direkt módon - egy számmal - megadjuk, hogy az illetô tömb hányadik elemét kívánjuk elérni.

Igy például, az : *int elemek[10] ;* módon definiált tömbnek 10 eleme van : elemek[0] , elemek[1], . . . , elemek[9] ; de nincs 10-es indexü eleme.

Példa az indexelésre : elemek[3] = 20;
vagy : for(i=0; i<10; i++) elemek[i] = 0;

Dimenzió :

Az elôbb bemutatott egydimenziós tömbön kívül kétdimenziós tömböt is használhatunk. Az egydimenziós tömböt a matematikában megismert egy dimenziós vektorhoz hasonlíthatjuk, a kétdimenziós tömböt pedig a mátrixhoz.
Ezekután az egydimenziós tömb **definiálása** úgy történik, hogy megadjuk a típusát, ezután - egy vagy több szóközt követôen - a nevét, majd szögletes zárójelek között, az elemek számát. A deklarálást követheti - kapcsos zárójelek között - a tömb elemeinek felsorolása, vesszôvel elválasztva. Ezt nevezzük a tömb inicializálásának ! Amennyiben a felsorolt elemek száma kevesebb a tömb méreténél, úgy a maradék elemek értéke nulla lesz !

Példa :

```
int vv[10] ;          // Egy integer típusú, vv nevû, 10 elemû tömb.
```

vagy :

```
int zz[10] = { 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 } ;

// Egy integer típusú, zz nevû, 10 elemû tömb, inicializálással.
```

A példákban szereplô deklarációk egyben definíciók is, mert a tömbméretnek megfelelô memóriaterület lefoglalódik !

5.1 Egydimenziós tömbök

1. sz. példaprogram :

```
/* ============ Legkisebb elem kiválasztása =========== */

 #include <stdio.h>

 void main()
 {
        //a egy 10 elemû,integer tömb,inicializálva.
     int a[10] = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };
     int i, min;

     min = a[0];
     for(i=0; i<10; i++)
     {
            // Ha az aktuális minimum nagyobb a
            // tömbelemnél
         if( min > a[i] )
               min = a[i] ;
     }
     printf("\nA legkisebb elem = %2d",min);
 }
```

Egydimenziós tömbbe történô adatbeolvasás a scanf() függvény segítségével oldható meg, a következô módon :

```
#include <stdio.h>

void main()
{
   int beolv[10], i;

   for(i=0;  i<10;  i++)  scanf("%d",&beolv[i]);
   . . .
}
```

Az "&" jel (cím operátor) ugyanúgy szükséges a tömbelem elôtt, mint egyszerû változó esetén ! (Lásd késôbb részletesen !)

5.2 Két dimenziós tömbök

A kétdimenziós tömb definiálása a típust és a nevet tekintve megegyezik az egydimenziós tömb definiálásával. A különbség a méretek számában van. A név után azt kell megadni, hogy a tömbnek hány sora van, utána pedig azt, hogy hány oszlopa. A kétdimenziós tömb tehát egy olyan vektor, amely azonos típusú és azonos elemszámú vektorokból áll ! **Az indexelés itt is 0-tól kezdôdik !**

Példa : A double x[4][5]; egy olyan kétdimenziós tömböt jelent amelynek 4 sora van (0...3) és 5 oszlopa (0...4) . Az x mátrix egy elemére kettôs indexeléssel hivatkozhatunk : x[2][3] = 5.67;. Az x mátrix 2-es indexü (azaz 3-ik) sora ill. 3-as indexü (azaz 4-ik) oszlopbeli eleme vegye fel az 5.67-es értéket. A mátrix tárolása a memóriában sorfolytonosan történik, azaz a 0. sor után az 1. sor következik stb. Az általunk hivatkozott x[2][3] tehát a memóriában a 2*5+3 = 13. elem. Az indexelés 0-tól indult !

A kétdimenziós tömbök inicializálása is megoldható : a tömb elemeit - kapcsos zárójelek között - soronként kell feltüntetni, vesszôvel elválasztva :

```
int mx[3][3] =    {
                     1, 2, 3,          // Egy integer típusú, mx nevû mátrix,
                     4, 5, 6,          // 3 sorral és 3 oszloppal, a megfelelô
                     7, 8, 9           // elemekkel inicializálva.
                  };
```

2. sz. példaprogram :

```
/* ======== Két double típusú  mátrix összege ======= */

   #include <stdio.h>
   #include <stdlib.h>
   main()
   {
      double a[3][3], b[3][3], c[3][3];
      int i, j;

      for(i=0; i<3; i++)  // Beolvasás az a mátrixba
      {
         for(j=0; j<3; j++)
         {
            printf("A[%d][%d] = ",i,j) ;
            scanf("%lf",&a[i][j]) ;
         }
      }
      printf("\n------------------------------\n\n");

      for(i=0; i<3; i++)   // Beolvasás a b mátrixba
      {
         for(j=0; j<3; j++)
```

```
         {
            printf("B[%d][%d] = ",i,j) ;
            scanf("%lf",&b[i][j]) ;
         }
      }
      printf("\n--------------------------------\n\n");

      for(i=0; i<3; i++)         // A c = a + b képzése
      {
         for(j=0; j<3; j++)
            c[i][j] = a[i][j] + b[i][j];

      }

      for(i=0; i<3; i++)         // A c mátrix kiiratása
      {
         for(j=0; j<3; j++)
           printf("%8.2lf%c",
            c[i][j],( ((j+1)%3 == 0) '\n' : ' ') );


      }
      system(“PAUSE”);
   }
```