

Tömbök definiálása

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2019. szeptember 21.

Azonos típusú elemekből álló adathalmazt akarunk létrehozni és rajta műveleteket végrehajtani.

A **tömb** (**array**) olyan objektumok halmaza:

- azonos típusúak
- memóriában folytonosan helyezkednek el
- az egyes elemek elérése egy sorszám (index) segítségével történik

Egydimenziós tömb: [vektor](#)

Azonos típusú elemekből álló adathalmazt akarunk létrehozni és rajta műveleteket végrehajtani.

A **tömb** (**array**) olyan objektumok halmaza:

- azonos típusúak
- memóriában folytonosan helyezkednek el
- az egyes elemek elérése egy sorszám (index) segítségével történik

Egydimenziós tömb: **vektor**

Hasonlóan a változókhoz, a tömböket is deklarálni kell:

```
típus vektornév[méret];
```

ahol a **méret** fordító által kiszámítható konstans kell legyen

Példa vektorok használatára

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define limit 5
5
6 int main()
7 {   double szamok[limit]=
8     {2.1, 4.3, 0.5, 11.2, 13};
9     double atlag = 0.0;
10    int i;
11
12    for (i=0; i< limit; i++)
13    {
14        atlag += szamok[i];
15    }
16
17    atlag = atlag/limit;
18
19    printf("Az atlaga = %f\n", atlag);
20
21    return 0;
22 }
```

- Definiálhatunk **makrókat**, ezek az előfordítás során behelyettesítésre kerülnek a program megfelelő helyére

Példa vektorok használatára

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define limit 5
5
6 int main()
7 {   double szamok[limit]=
8     {2.1, 4.3, 0.5, 11.2, 13};
9     double atlag = 0.0;
10    int i;
11
12    for (i=0; i< limit; i++)
13    {
14        atlag += szamok[i];
15    }
16
17    atlag = atlag/limit;
18
19    printf("Az atlaga = %f\n", atlag);
20
21    return 0;
22 }
```

- Definiálhatunk **makrókat**, ezek az előfordítás során behelyettesítésre kerülnek a program megfelelő helyére
- deklaráljuk **szamok** nevű, 5 elemű vektort és feltöltjük számokkal

Példa vektorok használatára

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define limit 5
5
6 int main()
7 {   double szamok[limit]=
8     {2.1, 4.3, 0.5, 11.2, 13};
9     double atlag = 0.0;
10    int i;
11
12    for (i=0; i< limit; i++)
13    {
14        atlag += szamok[i];
15    }
16
17    atlag = atlag/limit;
18
19    printf("Az atlaga = %f\n", atlag);
20
21    return 0;
22 }
```

- Definiálhatunk **makrókat**, ezek az előfordítás során behelyettesítésre kerülnek a program megfelelő helyére
- deklaráljuk **szamok** nevű, 5 elemű vektort és feltöltjük számokkal
- **for** ciklussal végigvesszük a vektor minden elemét

Példa vektorok használatára

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define limit 5
5
6 int main()
7 {   double szamok[limit]=
8     {2.1, 4.3, 0.5, 11.2, 13};
9     double atlag = 0.0;
10    int i;
11
12    for (i=0; i< limit; i++)
13    {
14        atlag += szamok[i];
15    }
16
17    atlag = atlag/limit;
18
19    printf("Az atlaga = %f\n", atlag);
20
21    return 0;
22 }
```

- Definiálhatunk **makrókat**, ezek az előfordítás során behelyettesítésre kerülnek a program megfelelő helyére
- deklaráljuk **szamok** nevű, 5 elemű vektort és feltöltjük számokkal
- **for** ciklussal végigvesszük a vektor minden elemét
- összeadjuk a **szamok** elemeit és tároljuk a **atlag** változóban

Példa vektorok használatára

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define limit 5
5
6 int main()
7 {   double szamok[limit]=
8     {2.1, 4.3, 0.5, 11.2, 13};
9     double atlag = 0.0;
10    int i;
11
12    for (i=0; i< limit; i++)
13    {
14        atlag += szamok[i];
15    }
16
17    atlag = atlag/limit;
18
19    printf("Az atlaga = %f\n", atlag);
20
21    return 0;
22 }
```

- Definiálhatunk **makrókat**, ezek az előfordítás során behelyettesítésre kerülnek a program megfelelő helyére
- deklaráljuk **szamok** nevű, 5 elemű vektort és feltöltjük számokkal
- **for** ciklussal végigvesszük a vektor minden elemét
- összeadjuk a **szamok** elemeit és tároljuk a **atlag** változóban
- átlag számolás és kiírás

Többsdimenziós tömbök deklarációja hasonló a vektoroknál látotthoz:

```
1  típus tombnev[meret1][meret2]...[meretn]
```

Továbbiakban csak kétdimenziós tömböket tekintünk.

```
1  típus tombnev[meret1][meret2]
```

meret1: sorok száma

meret2: oszlopok száma

Többdimenziós tömbök deklarációja hasonló a vektoroknál látotthoz:

```
1  típus tombnev[meret1][meret2]...[meretn]
```

Továbbiakban csak kétdimenziós tömböket tekintünk.

```
1  típus tombnev[meret1][meret2]
```

meret1: sorok száma

meret2: oszlopok száma

Figyelem!

- `tombnev[meret1,meret2]` jelölés nem működik
- C-ben at elemek soronként haladva tárolódnak, fentről lefelé (row-major)
- ellentétben pl a FORTRAN-nal, ahol oszlopontként haladva, balról jobbra (column-major)

Mátrixok megadása

Példa konstans mátrix megadására:

```
1 double matrix[3][2]={ {2.0, 4.1}, \\ 0. sor
2                       {1.0, 4.0}, \\ 1. sor
3                       {6.4, 9.3}, \\ 2. sor
4                       }
```

Mátrixok megadása

Példa konstans mátrix megadására:

```
1 double matrix[3][2]={ {2.0, 4.1}, \\ 0. sor
2                       {1.0, 4.0}, \\ 1. sor
3                       {6.4, 9.3}, \\ 2. sor
4                       }
```

A mátrixok bejárása két **for** ciklussal történhet:

```
1
2 int main()
3
4 int i,j;
5 double matrix[3][2]={ {2.0, 4.1}, \\ 0. sor
6                       {1.0, 4.0}, \\ 1. sor
7                       {6.4, 9.3}, \\ 2. sor
8                       }
9
10 for(i=0; i<3; i++)
11 {
12     for(j=0; j<2; j++)
13     {
14         printf("%d %d %f",i,j,matrix[i][j]);
15     }
16     printf("\n");
17 }
```