Tömbök I

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2023 szeptember 26.

Tömbök

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

C++ tömb jellegű tárolók

A C++-ban használható fontosabb tömb jellegű tárolók:

- beépített tömb adattípus
  - fordítási időben ismert méretű kell hogy legyen
  - alapesetben nem túl sok elemre praktikus
  - nagyon hasonlóan működik, mint más programnyelvekben (C, Fortran)
- `std::array`
  - "átmenet" a beépített tömb adattípus és a `std::vector` között (később látjuk)
  - alapesetben nem túl sok elemre praktikus
- `std::vector`
  - futási időben is változhat a méret
  - bármekkora lehet, amíg belefér a memóriába

C (Fortran) stílusú tömb

Egyszerű példa:

```
#include <iostream>

using namespace std;

int main()
{
    const unsigned s=5;

    double szamok[s]=
    {2.1, 4.3, 0.5, 11.2, 13};

    double atlag = 0.0;

     for (int i=0; i < s; ++i)
     {
     atlag += szamok[i];
     }

     atlag = atlag/s;
     cout << "atlag= " << atlag << endl;

     return 0;
}
```

- alapesetben a változók értéke változhat
- ez hibalehetőséget jelenthet (véletlenül felülírjuk az értéket)
- használjuk a `const` minősítőt

C (Fortran) stílusú tömb

Egyszerű példa:

```
#include <iostream>

using namespace std;

int main()
{
    const unsigned s=5;

    double szamok[s]=
    {2.1, 4.3, 0.5, 11.2, 13};

    double atlag = 0.0;

     for (int i=0; i < s; ++i)
     {
     atlag += szamok[i];
     }

     atlag = atlag/s;
     cout << "atlag= " << atlag << endl;

     return 0;
}
```

- alapesetben a változók értéke változhat
- ez hibalehetőséget jelenthet (véletlenül felülírjuk az értéket)
- használjuk a `const` minősítőt
- definiáljuk **szamok** nevű, 5 elemű vektort és feltöltjük számokkal

C (Fortran) stílusú tömb

Egyszerű példa:

```
#include <iostream>

using namespace std;

int main()
{
    const unsigned s=5;

    double szamok[s]=
    {2.1, 4.3, 0.5, 11.2, 13};

    double atlag = 0.0;

     for (int i=0; i < s; ++i)
     {
     atlag += szamok[i];
     }

     atlag = atlag/s;
     cout << "atlag= " << atlag << endl;

     return 0;
}
```

- alapesetben a változók értéke változhat
- ez hibalehetőséget jelenthet (véletlenül felülírjuk az értéket)
- használjuk a `const` minősítőt
- definiáljuk **szamok** nevű, 5 elemű vektort és feltöltjük számokkal
- `for` ciklussal végigvesszük a vektor elemeit. Figyelem! i=0..4

C (Fortran) stílusú tömb

Egyszerű példa:

```
#include <iostream>

using namespace std;

int main()
{
    const unsigned s=5;

    double szamok[s]=
    {2.1, 4.3, 0.5, 11.2, 13};

    double atlag = 0.0;

     for (int i=0; i < s; ++i)
     {
     atlag += szamok[i];
     }

     atlag = atlag/s;
     cout << "atlag= " << atlag << endl;

     return 0;
}
```

- alapesetben a változók értéke változhat
- ez hibalehetőséget jelenthet (véletlenül felülírjuk az értéket)
- használjuk a `const` minősítőt
- definiáljuk **szamok** nevű, 5 elemű vektort és feltöltjük számokkal
- `for` ciklussal végigvesszük a vektor elemeit. Figyelem! i=0..4
- összeadjuk a **szamok** elemeit és tároljuk a **atlag** változóban
- átlag számolás és kiírás

Többdimenziós tömbök

Többdimenziós tömbök deklarációja hasonló:

```
tipus tombnev[meret1][meret2];
```

- ez lényegében egy olyan tömb, amely `meret1` dimenziós, és
- minden egyes eleme egy másik tömb, amely `meret2` dimenziós
- lényegében tehát `meret1`: sorok száma; `meret2`: oszlopok száma

Többdimenziós tömbök

Többdimenziós tömbök deklarációja hasonló:

```
tipus tombnev[meret1][meret2];
```

- ez lényegében egy olyan tömb, amely `meret1` dimenziós, és
- minden egyes eleme egy másik tömb, amely `meret2` dimenziós
- lényegében tehát `meret1`: sorok száma; `meret2`: oszlopok száma

Példa konstans mátrix megadására:

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

Többdimenziós tömbök

Többdimenziós tömbök deklarációja hasonló:

```
tipus tombnev[meret1][meret2];
```

- ez lényegében egy olyan tömb, amely `meret1` dimenziós, és
- minden egyes eleme egy másik tömb, amely `meret2` dimenziós
- lényegében tehát `meret1`: sorok száma; `meret2`: oszlopok száma

Példa konstans mátrix megadására:

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

A mátrixok bejárása két `for` ciklussal történhet:

```
for(int i=0; i<3; ++i)
  {for(int j=0; j<2; ++j)
    {cout << matrix[i][j] << endl;}
    cout << "\n";
  }
```

Ajánlás

A C (Fortran) stílusú tömbök helyett használjuk a következőkben tárgyalt `std::vector` vagy `std::array` osztályt!