Mátrixok

Répásné Babucs Hajnalka

Egydimenziós tömb

```
int[] vektor;
vektor = new int[30];
int[] vektor2 = {1,2,3,4,5};
```

```
for (int i=0; i<vektor.Length;i++)
{       Console.WriteLine("A vektor2 {0}.-ik eleme: {1}", i, vektor[i]);    }

foreach (int i in vektor2)
{     Console.WriteLine("A vektor2 eleme: {0}", i);   }
```

Mátrix

```
int[,] matrix = new int[3,6];
int[,] matrix2= new int[,] {{1, 2, 3},{4, 5, 6}};

Random r = new Random();
for(int i = 0; i < matrix.GetLength(0); i++)
{
    for(int j = 0; j < matrix.GetLength(1); j++)
    {    matrix[i, j] = r.Next();    }
}
```

m[0,0]	m[0,1]	m[0,2]	…	m[0,m-1]
m[1,0]	m[1,1]	m[1,2]	…	m[1,m-1]
m[2,0]	m[2,1]	**m[2,2]**	…	m[2,m-1]
…	…	…	…	…
m[n-1,0]	m[n-1,1]	m[n-1,2]	…	m[n-1,m-1]

Jagged (egyenetlen) tömb

```
int[][] j_tomb= new int[3][];
Random r = new Random();
```

```
for(int i = 0; i < j_tomb.Length; i++)
{
        j_tomb[i] = new int[r.Next(1, 5)];
        for(int j = 0; j < j_tomb[i].Length; j++)
        {       j_tomb[i][j] = r.Next(100);                    }
}
```

Jagged tömb inicializálása

```
int[][] j_tomb2 = new int[][]
{
        new int[] {1, 2},
        new int[] {3, 4, 5}
};
```

Jagged tömb bejárása - for

```
for (int i = 0; i < j_tomb2.Length; i++)
{
    for (int j = 0; j < j_tomb2[i].Length; j++)
    {
        Console.WriteLine("Az {0}. sor {1}. eleme: {2}", i, j, matrix2[i, j]);
     }
}
```

Jagged tömb bejárása - foreach

```
foreach (int[] i in j_tomb2)
{
    foreach (int j in i)
    {
        Console.Write("{0} ",j);
    }
    Console.WriteLine();
}
```

Programozási tételek mátrixra - Összegzés

- Összegzés tétele:
  - Adott a **matr[0..n-1,0..m-1]** mátrix, amelynek elemein értelmezhető az összeadás művelete. Számítsuk ki a mátrix elemeinek összegét (**ossz** változó)!
- Algoritmus:

```
ossz := 0
ciklus i:=0-tól n-1-ig
  ciklus j:=0-tól m-1-ig
            ossz := ossz + matr[i,j]
  ciklus vége.
ciklus vége.
```

- Átlagszámításnál **n*m**-mel kell osztani!

Programozási tételek mátrixra - Megszámlálás

- Megszámlálás tétele:
  - Adott a **matr[0..n-1,0..m-1]** mátrix, amelynek elemein értelmezhető az adott tulajdonságfüggvény. Számoljuk meg, hogy a mátrixban hány darab, az adott tulajdonságnak megfelelő elem található (**db** változó)!

- Algoritmus:

```
db := 0
ciklus i:=0-tól n-1-ig
  ciklus j:=0-tól m-1-ig
            ha matr[i,j] adott tulajdonságú
            akkor db := db + 1
  ciklus vége.
ciklus vége.
```

Programozási tételek mátrixra – Szélsőérték kiválasztás

- Minimum- és maximumkiválasztás tétele:
  - Adott a **`matr[0..n-1,0..m-1]`** mátrix, amelynek elemein értelmezhető a kisebb / nagyobb reláció. Válasszuk ki a mátrix legkisebb / legnagyobb elemét! A **`mini`** és a **`minj`** változók jelentsék a szélsőérték helyét, azaz **sor- és oszlopindexét**!
- Algoritmus:

```
mini := 0
minj := 0
ciklus i:=0-tól n-1-ig
  ciklus j:=0-tól m-1-ig
            ha matr[i,j] < matr[mini,minj] akkor
                  mini := i
                  minj := j
  ciklus vége.
ciklus vége.
```

Programozási tételek mátrixra – Lineáris keresés

- Lineáris keresés tétele:
  - Adott a **`matr[0..n-1,0..m-1]`** mátrix, amelynek elemein értelmezhető az adott tulajdonságfüggvény. Keressük meg (sorfolytonosan) a mátrix első adott tulajdonságú elemét! Az **`l`** logikai változó legyen igaz, ha **sikerült** találni ilyen elemet, és ekkor az **`i`** és **`j`** tartalmazza az első adott tulajdonságú elem **sor- és oszlopindexét**!
- Algoritmus:

```
l := hamis
i := -1
ciklus amíg (i<n-1) és (NEM l)
  i := i+1
  j := -1
  ciklus amíg (j<m-1) és (NEM l)
       j := j+1
       ha matr[i,j] adott tulajdonságú akkor l:=igaz
  ciklus vége.
ciklus vége.
```