Fájlkezelés

Text fájl

- Sorokra tagolt, a sorok stringek.
- Minden sor végét speciális sorvége jel zárja: Windowsban \r\n karakterek, Linux alatt a \n (ha környezetfüggetlen programot akarsz: `Environment.NewLine`)
- A text fájl kezeléshez szükséges dolgok a **`System.IO`** névtérben vannak, ezért a **`using System.IO;`** sort bele kell írni a programunk elejére!
- Hogy a fájloknál ne kelljen elérési útvonalat adni, a txt állományokat érdemes az exe mellett elhelyezni. (A projekt könyvtárán belül **`bin\Debug`**)

Text fájl írása

Az íráshoz a **StreamWriter**-t kell használni:

```
string fnev = "adatok.txt";
StreamWriter w = new StreamWriter(fnev, false, Encoding.Default);
```

- Az első paraméter (*fnev*) adja meg a fájl nevét, amelybe írni szeretnénk.
  - Amennyiben teljes elérési utat szeretnénk használni (nem ajánlott, mert a program hordozhatóságának elvesztéséhez vezethet), akkor a \ jeleket duplázni kell, vagy az idéző jel elé @ jelet rakni.
  - Pl.: `string fnev = @"c:\adatok.txt";`
- A második paraméter (*`false`*) azt adja meg, hogy amennyiben ilyen nevű fájl már létezik, akkor azt a rendszer törölje le, és üres állapotból induljon a fájlba írás. Ha ide *`true`*-t adunk meg, akkor a már létező fájl tartalma megmarad, és amit most írunk bele majd a fájlba, az annak a végére fog íródni (hozzáfűzés, append).
- A harmadik paraméter a fájlba írás kódlapját adja meg. Az *`Encoding.Default`* a Windowsunkban használt kódlap (alapértelmezett kódlap)

A fentiek után a fájlba írás máris indulhat. Az íráshoz a *`WriteLine`* műveletet kell használni.

```
w.WriteLine("Fájlba írni jó");
double r = 23.4;
w.WriteLine("Kerület={0}", 2 * r * Math.PI);
```

Ha minden adatot beleírtunk a fájlba, akkor a végén le kell azt zárnunk a *`Close()`* hívásával:

```
w.Close();
```

Amennyiben elfelejtjük lezárni a fájlt, úgy a kiírt adatok nem kerülnek bele! Ezért ezt a lépést semmiképp ne felejtsük el!

Text fájlból olvasás

A fájlból olvasáshoz a **StreamReader**-t kell használni.

```
string fnev = "adatok.txt";
StreamReader r = new StreamReader(fnev, Encoding.Default);
```

A megnyitás során megadott paraméterek:

- A fájl neve (*fnev*), ez minden esetben kötelező.
- Ezen kívül meg lehet adni a fájlba íráskor használt kódlapot (ez most is a Windows alapértelmezett kódlapja lesz, *Encoding.Default*)

A fájlból olvasás során a *ReadLine()*-t kell használni. A fájlunkat most az *r* változó képviseli, ezért ***r.ReadLine()***-nal kell olvasni. A *ReadLine()* művelet string-et ad meg. Minden egyes olvasás során a fájlunk egy-egy sorát kapjuk meg.

```
string s;
s = r.ReadLine();
```

Amikor befejezzük a fájlból olvasást, akkor a fájlt le kell zárni a *Close()* segítségével:

```
r.Close();
```

Gyakori eset, amikor nem tudjuk előre, hogy a text fájlunk hány sorból áll. Ekkor *for* ciklus nem használható, helyette *while* ciklust alkalmazunk. A feltételben azt kérdezzük le, hogy elértük-e már (end of file, end of stream) a text fájl végét.

```
using System.IO;

string fnev = "adatok.txt";
StreamReader r = new StreamReader(fnev, Encoding.Default);
while (!r.EndOfStream)
{
    string s = r.ReadLine();
    …
}
r.Close();
```

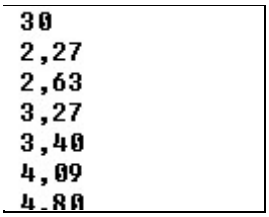
Split

A *Split* segítségével egy sztringet több felé vághatunk szét valamilyen elválasztó karakter mentén. A kettévágás egy string[] tömböt eredményez.

```
string[] felbontott = osszessor[i].Split(' ');
```

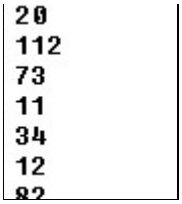
Példák beolvasásra:

1. Az *atlag.txt* első sorában egy szám található, utána pedig soronként annyi valós szám, amennyi az első érték volt:




```
StreamReader r1 = new StreamReader("atlag.txt");
//Fájl 1. sorában az adatok száma
int db1 = int.Parse(r1.ReadLine());
double[] tomb1 = new double[db1];
for (int i = 0; i < db1; i++)
{
    tomb1[i] = double.Parse(r1.ReadLine());
}
r1.Close();
```

2. A *szamok1.txt*-ben nincs megadva, hogy pontosan hány érték lesz, csak az, hogy maximum 50 egész számot tartalmaz a fájl.

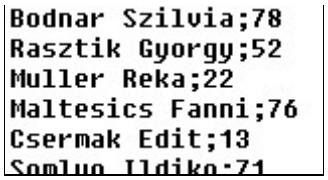



```
//Nincs megadva, hogy pontosan hány érték lesz, csak a maximális száma
StreamReader r2 = new StreamReader("szamok1.txt");
int n2 = 0; //Ennyi elemet tartalmazott a fájl
int[] tomb2 = new int[50]; //A feladat szerint max. 50 szám van
while (!r2.EndOfStream && n2<50)
{
    tomb2[n2] = int.Parse(r2.ReadLine());
    n2++;
}
r2.Close();
```

3. A *doli.txt* állományban több összetartozó adat van, de külön sorokban.


```
//Több, összetartozó adatom van, de külön sorokban
StreamReader r3 = new StreamReader("doli.txt", Encoding.Default);
int n3 = 0;
string[] nev = new string[1000];
int[] pontszam = new int[1000];
while (!r3.EndOfStream && n3 < 1000)
{
    nev[n3] = r3.ReadLine();
    pontszam[n3] = int.Parse(r3.ReadLine());
    n3++;
}
r3.Close();
```

4. Az *adatok.txt*-ben több összetartozó adat van egy sorban, az adatok egymástól ;-vel vannak elválasztva.




```
//Több, összetartozó adatom van, egy sorban
//de valamilyen speciális jellel elválasztva
StreamReader r4 = new StreamReader("adatok.txt");
int n4 = 0;
string[] nev4 = new string[1000];
int[] pontszam4 = new int[1000];
while (!r4.EndOfStream && n4 < 1000)
{
    string[] sv = r4.ReadLine().Split(';');
    nev4[n4] = sv[0];
    pontszam4[n4] = int.Parse(sv[1]);
    n4++;
}
r4.Close();
```