

Szöveges állományok kezelése

Készítette: Vastag Atila

2017

Az IO osztályok a *System.IO* névtérben vannak.

```
using System.IO;
```

A C# ún. *stream*-eket, adatfolyamokat használ az IO műveletek végzéséhez.

FileStream

StreamReader

StreamWriter

Kezdjük egy file megnyitásával és tartalmának a képernyőre írásával, és e file tartalma legyen a következő:

10.0.0.1
10.0.0.2
10.0.0.3
10.0.0.4
10.0.0.5
10.0.0.6

```
using System.IO;

namespace IP
{
    class Program
    {
        private static List<string> ipcimek = new List<string>();

        static void Main(string[] args)
        {
            FileStream fajl = new FileStream(@"adat.txt", FileMode.Open);
            StreamReader olvaso = new StreamReader(fajl);
            while (!olvaso.EndOfStream)
            {
                ipcimek.Add(olvaso.ReadLine());
            }
            olvaso.Close();
            fajl.Close();
        }
    }
}
```

A fájlhoz való hozzáféréshez létrehozzuk a file objektumot a **FileStream** osztályból. A létrehozáskor meg kell adnunk a fájl nevét és a fájllal végezni kívánt műveletet.

```
FileStream fajl = new FileStream(@"adat.txt", FileMode.Open);
```

```
FileStream fajl = new FileStream("d:\\adat.txt", FileMode.Open);
```

A fájl nevének megadásakor ügyeljünk arra, hogy az elérési útnál a C# a „\” karaktert különlegesen kezeli. Ezért vagy a @ jelet használjuk az elérési út megadása előtt, vagy pedig „\\” jelethasználunk.

A második paraméter a fájllal elvégzendő műveleteket határozza meg. A **FileMode** paraméter lehetséges értékeit az alábbi táblázat tartalmazza.

Paraméter	Leírás
<i>Append</i>	Ha a fájl nem létezik, akkor létrehozza. Ha létezik a fájl, akkor megnyitja, és a végére írja ki az új adatokat.
<i>Create</i>	Létrehoz egy üres fájlt. Ha már létezett az adott néven a fájl, akkor felülírja.
<i>CreateNew</i>	Létrehoz egy üres fájlt. Ha már létezett az adott néven a fájl, akkor egy kivételt generál.
<i>Open</i>	Megnyitja a fájlt. Ha a fájl nem létezik, akkor hibát generál.
<i>OpenCreate</i>	Megnyitja a fájlt. Ha nem létezik, akkor létrehozza.
<i>Truncate</i>	Megnyitja a fájlt, majd 0 hosszúságúra csonkítja.

A `FileStream` objektum létrehozásakor egy harmadik paramétert is használhatunk. Ennek segítségével a fájlhoz való hozzáférési jogosultságot adhatjuk meg. A *`FileAccess.Read` csak olvasási jogot, a `FileAccess.Write` csak írási jogot, a `FileAccess.ReadWrite` olvasási és írási jogot kér a fájlra.*

A szövegfájl kezelését végző `StreamReader` osztály példányosításakor paraméterként a fájl nevét kell megadnunk. Ha szükséges, akkor második paraméterként a szövegfájl kódolását is megadhatjuk a *`CurrentEncoding` felsorolás értékeivel.*

Miután megnyitottuk a fájlt olvasásra, ki kell olvasni belőle az adatokat. Ezt a `StreamReader` objektum metódusaival tudjuk megtenni.


A `StreamReader Read()` metódus egy karaktert tud beolvasni, míg a `ReadLine()` metódusa egy egész sort. A metódusok függvényként a visszatérési értékükként adják meg a beolvasott tartalmat.

Amikor egy szövegfájlból beolvasunk adatokat, nem mindig tudjuk, hogy hány adatsorunk van a fájlban. Ilyenkor használhatjuk az *EndOfStream* tulajdonságot, amely igaz értéket ad vissza, ha a fájl végén vagyunk.

A fájlból az adatokat ciklus segítségével dolgozhatjuk fel. A ciklus lehet dinamikus, számlálós vagy akár *foreach* ciklus is.

```
while (!olvaso.EndOfStream)
{

}
```



A fájlműveletek befejezésekor a *StreamReader* és a *FileStream* objektumokat a *Close()* eljárással be kell zárhatjuk.

```
olvaso.Close();
fajl.Close();
```

A **StreamReader** osztály fontosabb metódusait és tulajdonságait az alábbi táblázat foglalja össze.

Metódus, tulajdonság	Leírás
<i>Read()</i>	Beolvas egy karaktert, és a következő karakterre lép.
<i>ReadLine()</i>	Beolvas egy sort, és a következő sorra lép. A beolvasott sort stringként adja vissza.
<i>Peek()</i>	Beolvas egy karaktert, de nem lép tovább. Ha nincs több karakter, akkor -1-et ad vissza.
<i>ReadToEnd()</i>	Az aktuális pozíciótól minden karaktert beolvas.
<i>Close()</i>	Lezárja a szövegfolyam olvasót.
<i>CurrentEncoding</i>	A szöveg kódolását adja meg.

Most írjunk is a fileba. Erre a feladatra a *StreamReader* helyett a *StreamWriter* osztályt fogjuk használni:

```
using System;
using System.IO;

namespace ConsoleApplication1
{
    class Program
    {
        static public void Main()
        {
            FileStream fs = new FileStream(@"adat.txt", FileMode.Write, FileAccess.Write,
            FileShare.None);

            StreamWriter sw = new StreamWriter(fs);

            Random r = new Random();
            for (int i = 0; i < 10; ++i)
            {
                sw.Write(r.Next());
                sw.Write(Environment.NewLine);
            }

            sw.Close();
            fs.Close();
        }
    }
}
```

a program generál 10 random számot és ezeket a számokat egy *adat.txt* szöveges állományba írja

íráskor a *StreamReader* helyett a *StreamWriter* objektumot kell használni

A *StreamWriter* objektumnak azt mondjuk, hogy írja bele az új random számot, majd utána írjon bele egy új sort

JSON

beolvasása

```
private static List<Player> FetchData(string filePath)
{
    T data = null;

    using (FileStream fs = new FileStream(filePath, FileMode.Open,
FileAccess.Read, FileShare.None))
    {
        using (StreamReader r = new StreamReader(fs, Encoding.UTF8))
        {
            string jsonString = r.ReadToEnd();
            data = JsonConvert.DeserializeObject<T>(jsonString);
        }
    }

    return data;
}
```

JSON létrehozása

```
public static void SaveData(string filePath = "./data.json")
{
    using (FileStream fs = new FileStream(filePath, FileMode.Create, FileAccess.Write,
    FileShare.None))
    {
        using (StreamWriter r = new StreamWriter(fs, Encoding.UTF8))
        {
            string json = JsonConvert.SerializeObject(T);
            r.Write(json);
        }
    }

    _data = FetchData("./data.json");
}
```

1 – Egy szöveges állományban, **eredmeny.txt**, az érettségizők pontjai vannak elmentve a következő módon, soronként és a sorokban tabulátorral elválasztva:

Virág 9,28

Jázmin 6,26

Feladatunk, hogy kikeressük a legtöbb és legkevesebb pontot elért érettségizőt és a **max.txt** illetve a **min.txt** állományokba írjuk bele őket. Egyes logikai egészeket alkotó műveleteket függvényekkel oldjuk meg.

2 – A programunk feladata, hogy hőmérsékletet mérjen reggel délben és este, majd ezeket a hőmérsékleteket a **meresek.txt** állományokba mentse a hét minden napján (napi három mérés).

A hőmérsékleteket **Random** számmal adjuk meg 0 és 40 közt!

Keressük ki a hét végén, miután megtörtént az ősz mérés, hogy mekkora volt az átlag hőmérséklet reggel, délben és este és az **atlag.txt** állományba mentjük el.

Egyes logikai egészeket alkotó műveleteket függvényekkel oldjuk meg.

3 – Egy **forrás.txt** állományban számoljuk meg a magán és mássalhangzókat, számokat és egyéb szimbólumokat, majd az eredményt írjuk az **eredmeny.txt** állományba.

Egyes logikai egészeket alkotó műveleteket függvényekkel oldjuk meg.

forrás.txt:

“Microsoft suspends the Intel Kaby Lake and Ryzen AMD processors for Windows 7 and 8. Unfortunately, the tech mogul has no plans in bringing the next-gen chips to the old operating systems in the future. In fact, just last week, Microsoft released the KB 4012982 error entitled "Your PC uses a processor that isn't supported on this version of Widows.”

<http://www.universityherald.com/articles/69674/20170317/microsoft-blocks-windows-7-8-updates-ryzen-amd-intel-kaby.htm>

4 – Egy szöveges dokumentumban e-mail címek találhatóak és a hozzájuk tartozó jelszó. Kérjük meg a felhasználót hogy adja meg az e-mail címét és a jelszót és ha ez megtalálható a szöveges dokumentumban léptessük be a rendszerbe és egy log.txt állományba írjuk be, hogy mikor és ki lépett be. Az e-mail cím bekérésénél ügyeljünk arra, hogy megfeleljen az e-mail cím formátumának.

A feladatot az OOP-al oldjuk megoldjuk meg.

5 – A **konyvek.txt** állományban az adatok a következő módon vannak tárolva:

- Vezetéknév (íróé),
 - Keresztnév (íróé),
 - SzületésiDátum,
 - Cím,
 - ISBN,
 - Kiadó,
 - KiadvásiÉv,
 - ár,
 - Téma,
 - Oldalszám,
 - Honorárium (amit a könyvért kapott az író)
- a) Írjuk ki a képernyőre az össz adatot
- b) Keressük ki az informatika témájú könyveket és mentjük el őket az **informatika.txt** állományba
- c) Az 1900.txt állományba mentjük el azokat a könyveket amelyek az 1900-as években íródtak
- d) Rendezzük az adatokat a könyvek oldalainak száma szerint csökkenő sorrendbe és a **sorbarakott.txt** állományba mentjük el.
- e) „kategoriak.txt” állományba mentse el a könyveket téma szerint. Például:
- Thriller:
- könyv1
 - könyv2
- Krimi:
- könyv1
 - könyv2

6 – A **roplabda.txt** állományban az adatok a következő módon vannak tárolva:

Név,
Magasság,
Poszt,
Nemzetiség,
Csapat,
Ország (ahol a csapat játszik)

- a) Írjuk ki a képernyőre az össz adatot
- b) Keressük ki az ütő játékosokat az **utok.txt** állományba
- c) A **csapattagok.txt** állományba mentjük a csapatokat és a hozzájuk tartozó játékosokat a következő formában:
Telekom Baku: Yelizaveta Mammadova, Yekaterina Gamova,
- d) Rendezzük a játékosokat magasság szerint növekvő sorrendbe és a **magaslatok.txt** állományba mentjük el.
- e) Mutassuk be a **nemzetisegek.txt** állományba, hogy mely nemzetiségek képviseltetik magukat a röplabdavilágban mint játékosok és milyen számban.
- f) **atlagnalmagasabbak.txt** állományba keressük azon játékosok nevét és magasságát akik magasabbak mint az „adatbázisban” szereplő játékosok átlagos magasságánál.
- g) állítsa növekvő sorrendbe a posztok szerint a játékosok össz magasságát
- h) egy szöveges állományba, „*alacsonyak.txt*” keresse ki a játékosok átlagmagasságától alacsonyabb játékosokat. Az állomány tartalmazza a játékosok nevét, magasságát és hogy mennyivel alacsonyabbak az átlagnál, 2 tizedes pontossággal.

7 – A **magyarvarosok.txt** állományban az adatok a következő módon vannak tárolva:

Nev (város neve),
Város típusa,
Megye név,
Járas,
Kistérség,
Népesség,
Terület

- a) Írjuk ki a képernyőre az össz adatot
- b) Keressük ki a megyeszékhely megyei jogú városokat és mentjük el a **megyejogugarosok.txt** állományba
- c) Az **nepesseg.txt** állományba mentjük el azokat a településeket és a hozzájuk tartozó adatokat, ahol a népesség 50.000 és 100.000 közt van
- d) Keressük ki azokat a településeket, melyek területei meghaladják az 200-at és a **nagyteruletek.txt** állományba mentjük el.
- e) Keressük ki Békés megye össz települését és a **bekes.txt** állományba mentjük el.
- f) **megyeteruletek.txt** állományba mentjük el a megye nevét és területének nagyságát.

8 – A **lotto.txt** állományban az adatok a következő módon vannak tárolva:

Név,

tippek

- a) Írjuk ki a képernyőre az össz adatot
- b) Random számok segítségével generáljuk le a napi 7 nyerő számot és írjuk őket egy szüveges állományba melynek az aktuális nap lesz a neve
- c) Keressük ki, van(ak)-e 7 találatos szelvény(ek), ha igen írjuk ki a nyertesek nevét a **nyertesek-{mai dátum}.txt** állományba.
- d) Keressük ki, hogy a befizetett játékosok hány találatot értek el, és mentjük el a **talalatok-{mai dátum}.txt** állományba a játékos nevét és a találatainak számát

9 – A **vezetok.txt** állományban az adatok a következő módon vannak tárolva:

Vezetéknév,Keresztnév,Anya vezetéke, Anya keresztnéve

Születés időpont,Születés helye,Megye,Ország

Utca,Házszám,Írányítószám,Város,Megye,Ország

Kategóriák

Az egyes egészek tabulátorral vannak elválasztva, az egészek adatai pedig vesszővel.

Írjunk programot, mely menü segítségével lehetővé teszi a következő adatok kérését:

- a) Írjuk ki a képernyőre az össz adatot a vezetőkről
- b) A felhasználó által megadott megyére a **megye-vezetoi.txt** állományba elmenti a megadott megyében lakó vezetőket.
- c) A felhasználó által megadott kategóriával rendelkező vezetőket a **{kategoria nev}-kategoria.txt** állományba menti el.
- d) A **fiatalok.txt** állományba kikeresi azokat a vezetőket akik 18 és 21 év között vannak.
- e) **kulfoldi.txt** állományba azokat a vezetőket keresi ki, akik nem Magyarországon születtek.

10 – Az **nb1.txt** állományban az adatok a következő módon vannak tárolva:

A labdarúgó mezére írt szám (szám)

A labdarúgó utóneve (szöveg); előfordul, hogy valaki felvett nevet használ, ilyenkor üres is lehet

A labdarúgó vezetéckneve (szöveg)

A labdarúgó születési dátuma (dátum)

Értéke igaz, ha magyar állampolgár (is) a labdarúgó (logikai) [-1 igaz, 0 hamis]

Értéke igaz, ha külföldi állampolgár (is) a labdarúgó (logikai) [-1 igaz, 0 hamis]

A labdarúgó euró ezrekben kifejezett értéke (szám)

A klub neve (szöveg)

A poszt neve (szöveg), például kapus, bal oldali védő, bal szélső

Írjunk programot, mely menü segítségével lehetővé teszi a következő adatok kérését:

a) A kapusokon kívül mindenkit mezőnyjátékosnak tekintünk. Keresse ki a legidősebb mezőnyjátékos vezetéck- és utónevét, valamint születési dátumát! (Feltételezheti, hogy csak egy ilyen játékos van.)

- b)** Határozza meg hány magyar, külföldi és kettős állampolgárságú játékos van!
- c)** Határozza meg játékosok összértékét csapatonként és írja ki a képernyőre! A csapatok neve és a játékosainak összértéke jelenjen meg!
- d)** Keresse ki, hogy mely csapatoknál mely posztokon van csupán egy szerződött játékos! Írja ki a csapat nevet és a posztot amire csak egy játékost szerződtek!
- e)** Keressük ki azon játékosokat, akiknek az értékük nem haladja meg a játékosok értékének átlag értékét.
- f)** Írja ki azon játékosok nevét, születési dátumát és csapataik nevét, akik 18 és 21 év közt vannak és magyar állampolgárok. Ha nincs ilyen, akkor megfelelő üzenettel helyettesítse a kimenetet.
- g)** A „hazai.txt” illetve a „legios.txt” állományokba keresse ki a magyar, illetve a külföldi állampolgárságú játékosokat csapatonként. A szöveges állományoknak tartalmazniuk kell a csapat nevét majd alatta felsorolva a játékosok teljes nevét, poszt nevet és értéküket.

11 - Adva van az adatok.txt állományban a Magyar Női Röplabda Bajnokság csapatainak pontszámai a következő képpen:

Békéscsaba
1,2,1,3,3,3,3,3,3,1,2,2,1,3,3,1,3

ahol a csapat nevét tabulátorral elválasztva követik a az elért pontok mérkőzésenként (max 18 lejátszott mérkőzés).

- a) Hány csapat vett részt a bajnokságban?
- b) Ki nyerte a bajnokságot?
- c) Döntetlen mérkőzéskor a csapat 2 pontot szerez. Mutassa be csapatonként ki hány döntetlen mérkőzést játszott le!
- d) Ha egy mérkőzés 5 szetben dől el, akkor a vesztes csapat 1 pontot szerez. Mely csapatok játszottak 5 szettes mérkőzést és hányat?
- e) Ki a bajnokság első három helyezete. Mutassa be mintának megfelelően:

<i>helyezés</i>	<i>- csapat neve</i>	<i>pontszám</i>
-----------------	----------------------	-----------------
- e) Az elért pontok alapján, az utolsó három csapat kiesik az első osztályból. Kik ők?
- f) Mutassa be csapatonként a győzelmi és verességi arányt csapatonként!
- g) Mely csapatok győzelmi aránya van az átlag alatt?

60 – Készítsünk chat szimulációs programot a magadott mintának megfelelően!

Kata: Szia!

Én: Helló!

Kata: Van házi?

Én: Mi bor? Az nincs. Megittam.

Kata: Mármint házi feladat.

Én: Ja, az nincs már, megcsináltam!

*A beszélgetés kezdetekor, a program ellenőrizze, hogy volt e korábbi chat a két fél közt. Ha igen, akkor a **Kata** - **Én**.txt fileban lesz megtalálható. Ilyen esetben a file tartalmát először írjuk ki a képernyőre, amjd folytatódhat a chat.*

A beszélgetés a végtelenig tartson. A chatből úgy lehet valakinek kilépni, hogy beír egy speciális karaktert, pl. @exit@. Attól, hogy a beszélgetés egyik tagja kilépett a chat nem szűnik meg.

Amikor mind a két fél kilépett, akkor a chat új tartalmát hozzá kell fűzni a már meglévő szöveges állományhoz, ha van ilyen, ha nincs, akkor újat kell létrehozni. Menteni kell azt, hogy ki írta, mit és mikor. A szöveges állomány minden egyes sora ezen adatokat kell, hogy tartalmazza tabulátorral elválasztva.