Fájlkezelés (internetről)

Fájlnak nevezünk minden háttértárolón található adatot, adathalmazt (pl. szövegszerkesztőben megírt dokumentum, stb.). A fájlok: azonos típusú komponensekből felépülő adatszerkezetek.

- a programba az adatokat nem csak billentyűzet vagy egér segítségével lehet bevinni, hanem valamilyen háttértárolón tárolt fájlból is be lehet olvasni, főleg nagy mennyiségű adat esetén
- hasonlóan, a program által előállított adatokat nem csak a képernyőre, hanem fájlba is ki lehet írni
- C#-ban a fájlok közös műveleteit stream-ekben valósították meg: a stream adatfolyam, a memória és egy külső egység közötti adatáramlás véghezvitelére
- mielőtt egy fájllal bármilyen műveletet végeznénk, meg kell azt nyitnunk. A fájl megnyitását követően különféle műveleteket végezhetünk. Ezek közül a legfontosabbak az olvasás és az írás. Olvasásnál a fájlból adatokat olvasunk be a memóriába, ami a gyakorlatban egy előzőleg deklarált változó feltöltését jelenti. Ezzel szemben írásnál a memóriából viszünk ki adatokat a fájlba. Az írást és olvasást összefoglalóan I/O (InputOutput) műveleteknek nevezzük
- a .NET számos osztályt biztosít a fájlkezelésre
- a fájl input-output szolgáltatásokat a system.IO névtér osztályai nyújtják

Állományfajták

- Szöveges fájlok – soros hozzáféréssel kezelhetjük
- Bináris fájlok – közvetlen hozzáféréssel kezelhetők: tetszőleges elemükre pozícionálhatunk, akkor is, ha ez az elem valahol a fájl belsejében helyezkedik el
- Xml – XmlTextReader és XmlTextWriter osztályok állnak a rendelkezésünkre (Az XML általános célú leírónyelv, amelynek elsődleges célja strukturált szöveg és információ megosztása az interneten keresztül.)

Szöveges fájlok kezelése

- a szöveges fájlokban levő adatok egyszerű szerkesztőprogramokkal (notepad) megtekinthetők
- a szöveges állományokban az adatok szövegformátumban vannak
- a szöveges formájú adatok sorokra vannak tördelve
- a szöveges fájl input-output műveletek osztályai a StreamReader és StreamWriter osztályok
- szükségünk lehet a System.Text névtérre (pl. a karakterkódolás beállításához)

Szöveges fájl megnyitása olvasásra

- text fájlok olvasásra való megnyitására a StreamReader osztályból kell példányosítani: a StreamReader osztály konstruktorának paraméterként meg kell adni a megnyitandó fájl nevét (kötelező) és meg lehet adni a kódlapot, amellyel a fájl tartalma íródni fog (alapértelmezésként a Default, amit a Windows aktuálisan is használ)
- ha a fájl nem a program saját könyvtárában van, akkor a teljes elérési utat is meg kell adni
- az elérési útvonalnál vigyázni kell, mert a backslash („\“) karakternek speciális jelentése van, ezért egyetlen backslash leírásához dupla „\\“ kell, vagy használjuk a @ jelet az elérési út előtt, amivel az utána következő sztring literál minden karakterét normális karakterként fogja értelmezni

  “C:\\Mappa1\\Mappa2\\proba.txt” vagy @”C:\Mappa1\Mappa2\proba.txt”

Pl.
```
StreamReader f = new StreamReader("c:\\proba.txt", Encoding.Default);
```

- előfordulhat, hogy a fájl tartalmának kiírásakor az ékezetes karakterek helyett kérdőjel jelenik meg. Ez azért van, mert az éppen aktuális karaktertábla nem tartalmazza ezeket a karaktereket, ez tipikusan nem magyar nyelvű operációs rendszer esetén fordul elő: megadjuk a kódlapot:

Pl.
```
StreamReader f = new StreamReader (@"c:\proba.txt",Encoding.GetEncoding("iso-88592"));
```

- szöveges fájl olvasásra való megnyitására használhatjuk a File osztály OpenText metódusát is

Pl.
```
StreamReader f = File.OpenText(@"D:\GME\tanszek-2013-2014\C#\pr1.txt");
```

Szöveges fájl olvasása

- miután megnyitottuk a fájlt olvasásra, utána olvashatjuk a tartalmát
- a textfájlokban az adatok sorokra vannak bontva
- egy lépésben általában egy sort szoktunk kiolvasni a **ReadLine()** metódussal

Pl.
```
StreamReader f = new StreamReader("c:\\proba.txt");
string s = f.ReadLine();
```

- minden ReadLine()-nal történő olvasási művelet során automatikusan lépünk a fájlban a következő sorra
- a ReadLine() többszöri alkalmazása révén eljutunk a fájl végéig

- karakterenként is kiolvashatjuk a fájl tartalmát a Read() függvénnyel, melynek visszatérési értéke a kiolvasott karakter kódja
- a Read()-del történő olvasáskor automatikusan lépünk a fájlban a következő karakterre

Pl.
```
char c = (char)f.Read();
```

Fájl végének elérése

- a fájlból való folyamatos olvasás esetén előbb-utóbb elérjük a fájl végét
- a fájl végének elérésekor a kiolvasott sztring null értéket vesz fel

Pl.
```
string s= f.ReadLine();
while(s!=null)
{
   Console.WriteLine(s);
   s=f.ReadLine();
}
```

vagy így:

```
string s;
while(!f.EndOfStream)
{
    s=f.ReadLine();
   Console.WriteLine(s);
}
```

Megj.: az `s` változó kiküszöbölhető

- fájl végének elérését le tudjuk kérdezni a Peek() metódus segítségével: a Peek() a soron következő bájt értékét adja meg anélkül, hogy a pozíciót léptetné; ha nincs következő bájt, vagyis elértük a fájl végét, akkor a Peek() által visszaadott érték -1

Pl.
```
string s;
while (f.Peek() != -1)
{
   s = f.ReadLine();
   Console.WriteLine(s);
}
```

Pozícionálni nem lehet a szöveges fájlban.

Szöveges fájl megnyitása írásra

- szöveges állományok írásra való megnyitására a StreamWriter osztályból kell példányosítani

- a StreamWriter osztály konstruktorának első paraméterként meg kell adni a fájl nevét az elérési útvonallal együtt, második paraméterként megadhatjuk a hozzáfűzés módját, ami lehet: false (ez az alapértelmezett) – létező fájl esetén törli a tartalmát, true – hozzáírásra nyitja meg a fájlt; harmadik paraméterként megadhatjuk a kódlapot, ugyanúgy, mint az olvasásnál

Pl.
```
StreamWriter f = new StreamWriter(@"C:\proba2.txt", true, Encoding.UTF8);
```

  - első paraméter: proba2.txt a fájl neve, amelybe írni szeretnénk
  - második paraméter: true – ha létezik a fájl, akkor annak tartalma megmarad, és amit most írunk bele, az annak a végére fog íródni (append)
  - harmadik paraméter: Encoding.Default – a fájlba írás kódlapját adja meg, ami jelen esetben a Windows-unkban használt kódlap

- a File osztály AppendText és CreateText metódusai is rendelkezésünkre állnak szöveges fájl létrehozására illetve létező fájl megnyitására

Pl.
```
StreamWriter f = File.AppendText(@"C:\proba.txt");
```

- megnyitja a fájlt írásra: ha létezett, az állomány végére pozícionál és íráskor oda ír, ha nem létezett, akkor létrehozza

Szöveges fájl írása

- miután megnyitottuk a fájlt írásra, a Write() és WriteLine() metódusok segítségével írhatunk bele
- a két metódus használata megegyezik a konzolos metódusokkal, csak most nem a konzolra, hanem az „f”-fel azonosított fájlba írunk

Pl.
```
StreamWriter f = new StreamWriter(@"C:\proba2.txt", true, Encoding.UTF8);
f.WriteLine(Console.ReadLine();
int t = 12; double u=6.23;
f.Write(t+" "+u);
```

- a WriteLine() a kiírás végén egy sorvége jelet is ír a fájlba, több WriteLine() használata esetén minden kiírás adatai új sorba kerülnek

File bezárása

- megnyitott fájlt a Close() metódussal tudunk bezárni
- amennyiben elfelejtünk bezárni egy írásra megnyitott fájlt, akkor bizonyos mennyiségű adatmódosítás elveszhet
  Pl. f.Close();
- mindenképpen ajánlott a bezárás előtt a puffert, vagyis az átmeneti tárolót üríteni
  Pl. f.Flush();