Szoftvertervezés és -fejlesztés I.

Szöveges fájlok kezelése

Felsorolástípus

Objektumtömb

Gyakorló feladatok

Hallgatói tájékoztató

A jelen bemutatóban található adatok, tudnivalók és információk a számonkérendő anyag vázlatát képezik. Ismeretük szükséges, de nem elégséges feltétele a sikeres zárthelyinek, illetve vizsgának.

Sikeres zárthelyihez, illetve vizsgához a jelen bemutató tartalmán felül a kötelező irodalomként megjelölt anyag, a gyakorlatokon szóban, illetve a táblán átadott tudnivalók ismerete, valamint a gyakorlatokon megoldott példák és az otthoni feldolgozás céljából kiadott feladatok önálló megoldásának képessége is szükséges.

Szoftvertervezés és -fejlesztés I.

Szöveges fájlok kezelése

Felsorolástípus

Objektumtömb

Gyakorló feladatok

Szöveges fájlok kezelése

- A szöveges fájlok egyes soraiban stringek találhatók
- A fájlokat soronként elölről hátrafelé haladva tudjuk bejárni
- A fájlt háromféle módon nyithatjuk meg
  - Olvasás: Ilyen esetben soronként ki tudjuk olvasni a fájl tartalmát
  - Írás: Ilyenkor új fájl jön létre, melybe soronként írhatunk, vagy a már létező fájl törlődik és egy ugyanolyan nevű új fájlba írhatunk soronként
  - Hozzáfűzés: Ekkor a fájlunk végére tudunk új sorokat írni
- A fájlkezelés végén a fájlt kötelező bezárnunk
- A fájlműveletek a `System.IO` névtérben találhatók

Fájl olvasása

- A StreamReader osztályt kell használnunk

```
StreamReader sr = new StreamReader("fajlnev.txt");
```

- Egy sor kiolvasása a fájlból

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

- Fájl végének ellenőrzése

```
sr.EndOfStream
```

- Fájl bezárása

```
sr.Close();
```

```
StreamReader sr = new StreamReader("fajlnev.txt");
while (!sr.EndOfStream)
{
    string s = sr.ReadLine();
    Console.WriteLine(s);
}
sr.Close();
```

Fájl írása

- A StreamWriter osztályt kell használnunk

```
StreamWriter sw = new StreamWriter("fajlnev.txt");
```

- Egy sor kiírása a fájlba

```
sw.WriteLine(s);
```

- Fájl bezárása

```
sw.Close();
```

```
StreamWriter sw = new StreamWriter("fajlnev.txt");
foreach (string s in sTomb)
{
    sw.WriteLine(s);
}
sw.Close();
```

Hozzáfűzés és karakterkódolás

- Ha hozzá akarunk fűzni már létező fájlhoz, akkor a fájl megnyitásakor kell ezt megadnunk

```
StreamWriter sw = new StreamWriter("fajlnev.txt", true);
```

- `false` használata felülírást eredményez

```
StreamWriter sw = new StreamWriter("fajlnev.txt", false);
```

- A szövegfájlok karakterkódolása a mentő programtól, beállítástól függhet
- A StreamWriter-t úgy kell megnyitni, hogy a szövegfájlénak megfelelő karakterkódolásban olvasson
  - Alapértelmezetten UTF-8-ban próbál olvasni
- StreamWriter megnyitása karakterkódolás megadásával:

```
StreamWriter sw =
    new StreamWriter("fajlnev.txt", Encoding.Default);
```

Elérési út megadása

- A fájlnév megadásánál megadhatunk abszolút elérési utat:

```
StreamWriter sw = new StreamWriter("c:\\hallgato\\fajlnev.txt");
```

- `StreamWriter sw = new StreamWriter(@"c:\hallgato\fajlnev.txt");`

- Nem ajánlott, más rendszeren nem lesz pontosan azon a helyen a fájl

- Relatív elérési út megadása:

- `StreamWriter sw = new StreamWriter("fajlnev.txt");`
- `StreamWriter sw = new StreamWriter("szovegek\\fajlnev.txt");`

- Ez a megadott nevű fájlt a munkakönyvtárban (working directory) keresi
- A munkakönyvtár egyszerű esetben az a könyvtár, ahol az exe van

Feladatok

1. Egy string tömbbe olvassa be egy szöveges fájl sorait, majd a sorokat írja ki a konzolra!
2. Határozza meg, hogy hány sora van a beolvasott fájlnak, illetve hány betű található benne.
   A fájl végére írjon két új sort, melyek ezeket az adatokat tartalmazzák!
3. Olvasson be egy fájlt, majd a beolvasott szöveget alakítsa át úgy, hogy csak a betűket és számokat tartsa meg. Az eredménnyel írja felül az eredeti fájlt!

Feladatok

4. Egy szöveges fájlt alakítson át úgy, hogy minden sor középre legyen rendezve!
5. Egy szöveges fájlt alakítson át úgy, hogy minden sor sorkizárt legyen!
6. Határozza meg, hogy egy szöveges fájlban melyik a leghosszabb szó, illetve melyik szó fordul elő leggyakrabban!

Szoftvertervezés és -fejlesztés I.

Szöveges fájlok kezelése

Felsorolástípus

Objektumtömb

Gyakorló feladatok

Felsorolás típus

- Olyan adatszerkezet, amely meghatározott értékek névvel ellátott halmazát képviseli.
- enum kulcsszóval definiáljuk
- Osztályon belül vagy kívül is definiálható

```
enum Napok
{
    Hétfő, Kedd, Szerda, Csütörtök, Péntek, Szombat, Vasárnap
}

class Program
{
    static void Main(string[] args)
    {
        Napok ma = Napok.Csütörtök;
        if (ma == Napok.Csütörtök)
            Console.WriteLine("Ma csütörtök van");
    }
}
```

Felsorolás típus

- A felsorolás típus elemeit számok reprezentálják. Kezdősorszám: 0
- Lehetőség van egyéni számozás használatára is

```
enum Napok
{
    Hétfő = 1, Szerda = 3, Csütörtök = 4
}

class Program
{
    static void Main(string[] args)
    {
        Napok ma = Napok.Csütörtök;
        int x = (int)ma; // x == 4
        Napok hétEleje = Napok.Hétfő;
        x = (int)hétEleje; // x == 1
    }
}
```

Szoftvertervezés és -fejlesztés I.

Szöveges fájlok kezelése

Felsorolástípus

Objektumtömb

Gyakorló feladatok

Objektumtömb

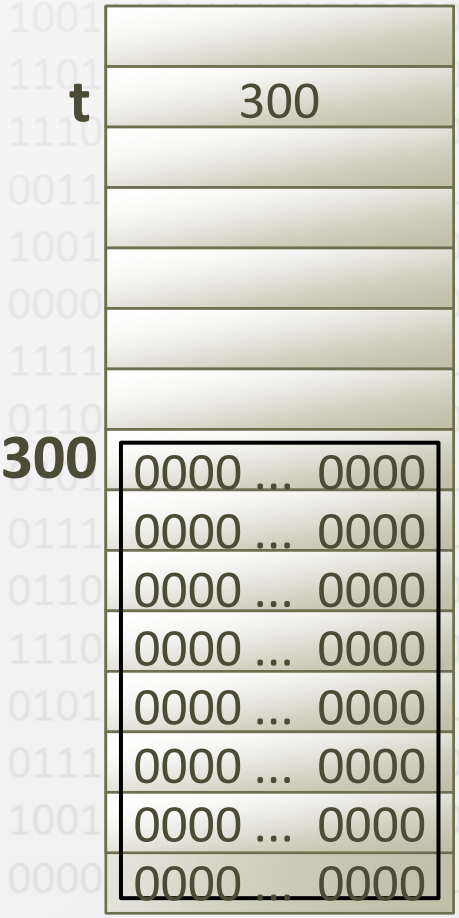
- **Ismétlés: tömblétrehozás**

```
int[] t = new int[8];
```

```
string[] t = new string[8];
```

```
char[] t = new char[8];
```

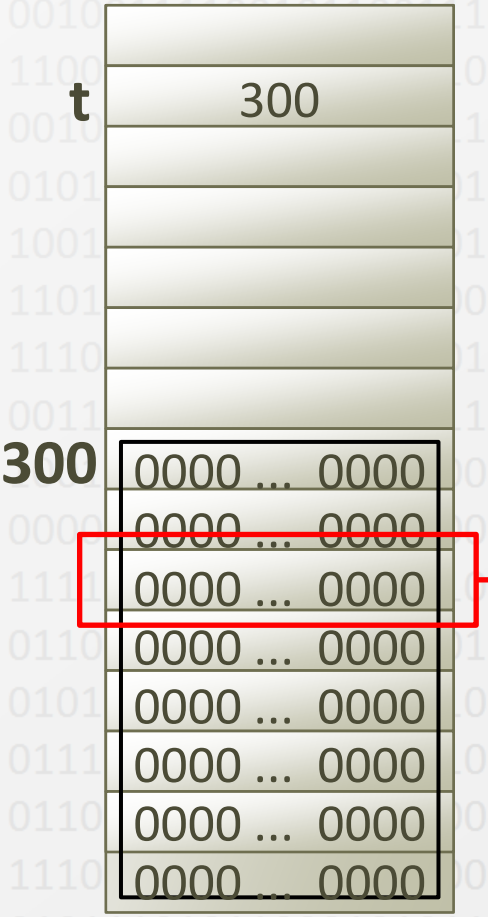
```
Osztaly[] t = new Osztaly[8];
```



- A tömb referenciatípus, így a változóban nem közvetlenül az adat helyezkedik el, hanem egy memóriacím (referencia)
- Az adott memóriacímen helyezkedik el az adat maga
- **Tömb létrehozásakor a tömb által lefoglalt adatterület kinullázódik a memóriában**

Objektumtömb

- **Ismétlés: alapértelmezett érték**



  - A tömbelemben a nullázás után a típushoz tartozó alapértelmezett érték van:
  - **Ha *érték típusú* változók tömbje volt, akkor közvetlenül az adat van az elemben:**
    - Ha a tömb int[] volt: 0
    - Ha a tömb char[] volt: \0 (a 0 kódú karakter)
    - Ha a tömb bool[] volt: false
  - **Ha viszont *referencia típusú* változók tömbje volt, akkor ez a csupa 0 valamilyen memóriacím (referencia)**
    - Nem érvényes memóriacím
    - **null**-nak nevezzük
    - **null** elemre nem lehet hivatkozni

Objektumtömb

- **Ismétlés: alapértelmezett érték**
  - Érték típusú változókat tartalmazó tömbök:

```
int[] t = new int[8];
```

```
char[] t = new char[8];
```

    - A tömbelemek értéke a létrehozás után érvényes, azonnal használható

```
int[] t = new int[8];
t[2]++;
Console.WriteLine(t[2]); //1
```

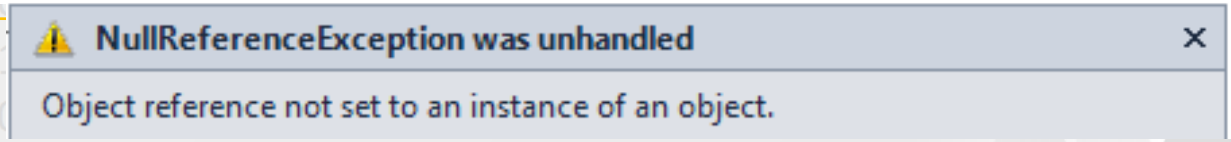
  - Referencia típusú változókat tartalmazó tömbök:

```
string[] t = new string[8];
```

```
Osztaly[] t = new Osztaly[8];
```

    - **A tömbelemek értéke a létrehozás után null, érvénytelen**
    - Ezért értéket kell adni minden tömbelemnek, mielőtt használatba vesszük

```
Osztaly[] t = new Osztaly[8];
Console.WriteLine(t[2].Tulajdonsag);  //NEM JÓ!!!
```

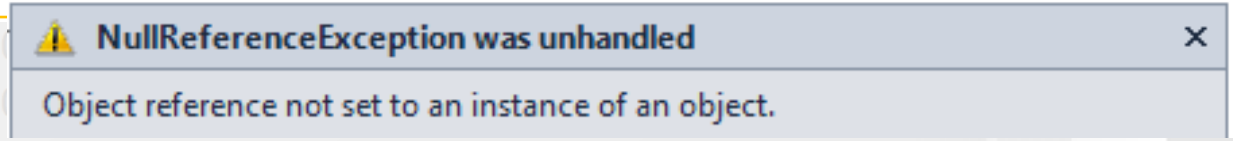


Objektumtömb

- **Objektumtömbben példányosítani kell minden elemet**

```
class Vektor
{
    private double x;
    public double X { get { return x; } set { x = value; } }
    private double y;
    public double Y { get { return y; } set { y = value; } }
    ...
    public Vektor(double x, double y)
    {
        this.x = x; this.y = y;
    }
}
```

```
Vektor[] tomb = new Vektor[8];
Console.WriteLine(tomb[2].X + " " + tomb[2].Y); //NEM JÓ!!!
```

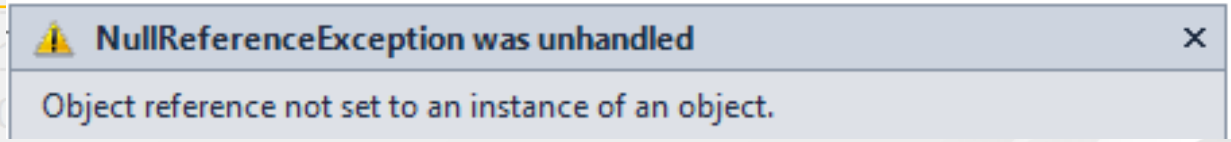



```
Vektor[] tomb = new Vektor[8];
for (int i = 0; i < tomb.Length; i++)
{
    tomb[i] = new Vektor(5, 5);
}
Console.WriteLine(tomb[2].X + " " + tomb[2].Y); //JÓ: 5 5-öt ír ki
```

Objektumtömb

- **A string is referenciatípus**
  - Speciális tulajdonsága miatt számos esetben érték típusként viselkedik
- **Ezért az alapértelmezett érték ott is null**
- **A stringtömbben is értéket kell adni minden elemnek:**

```
string[] t = new string[8];
Console.WriteLine(t[2].ToUpper()); // NEM JÓ!!!
```




```
string[] t = new string[8];
for (int i = 0; i < tomb.Length; i++)
{
     t[i] = "ffff";
}
Console.WriteLine(t[2].ToUpper()); // FFFF
```

Szoftvertervezés és -fejlesztés I.

Szöveges fájlok kezelése

Felsorolástípus

Objektumtömb

Gyakorló feladatok

Feladatok

1. **Írjon menüvezérelt programot, amely lehetővé teszi egy futó napi edzéseinek eltárolását**

   a) **Új edzés felvitele**
   - Dátum YYYYMMDD formátumban
   - Táv #,# km formában
   - Idő HH:MM:SS formátumban

   Az új edzés bekerül a futónapló fájl végére

   b) **Edzések listázása**

   A konzolra kilistázza az edzéseket

   c) **Statisztikák**

   A konzolra és egy statisztika fájlba kiírja az edzések számát, valamint a távok és idők összegét és átlagát

Gyakorló feladat

2. Csevegőprogramot készítünk, amelyben Személy típusú elemekkel reprezentáljuk a kontaktjainkat. Egy Személynek van neve, születési éve és neme.

a) Hozzunk létre 5 elemű tömböt, amelyet feltöltünk Személyekkel! A Személy osztályban legyen olyan paraméteres konstruktor, amelynek segítségével név és nem ismeretében, de véletlenszerű születési évvel létrehozhatjuk a személyt.

b) Vannak olyan emberek, akiknek ugyanaz a vezeték- vagy keresztnevük, mint a felhasználónak? Ha vannak, akkor listázzuk ki mindet.

c) Listázzuk ki az összes olyan Személyt, aki a felhasználóval ellenkező nemű!

d) Számoljuk meg, hány olyan Személy van, aki a felhasználóval azonos korosztályba tartozik (azaz maximum 5 év van közöttük)!

Feladatok

3. Készítsen szótárprogramot, amely szópárokat tárol, illetve ki is kérdezi a szavakat!

a) Lehessen új szópárokat felvenni, de csak akkor, ha még nincsenek bent a szótárban!

b) Lehessen kilistázni a szótár tartalmát!

c) Tudja kikérdezni a szavakat mindkét nyelv szavai alapján. A kikérdezés végén adja meg, hány helyes válasz érkezett!

Feladatok

4. Hozzunk létre egy 20 elemű tömböt, amelyben Zh típusú elemek vannak. Egy ilyen elemmel egy zh-eredményt reprezentálunk. Egy Zh-hoz egy 6 jegyű Neptun-kód és egy 0-100-ig terjedő pontszám tartozik. Egy konstruktor segítségével véletlenszerű Neptun-kóddal és pontszámmal hozzuk létre a tömbben lévő példányokat.

a) Sorolja fel, kik mentek át a zh-n!

b) Írja ki a legjobb eredményű hallgató Neptun-kódját! Ha több embernek is ez az eredménye, akkor mindegyikükét.

c) Listázza ki az eredményeket úgy, hogy a nevek mellett a jegyek szerepelnek! A jegyeket ponthatártömb alapján állapítsa meg.

A Neptun-kód egy string, amely kezdetben üres. Mind a 6 karakterét legeneráljuk egyenként, a következőképpen:
Véletlenszám sorsolásával eldöntjük, szám jön-e vagy karakter.
Ha szám, akkor sorsolunk egy számot 0-9-ig, és hozzáadjuk a stringhez.
Ha karakter, akkor sorsolunk egy karaktert, és azt hozzáadjuk a stringhez. Karakter sorsolása:

```
char c = (char)rand.Next(65, 91);
```