7. gyakorlat

C#/.NET alapú grafikus alkalmazás fejlesztés

A C#/.NET alapú grafikus alkalmazás fejlesztéshez kapcsolódó első gyakorlaton a C# nyelvvel ismerkedünk meg konzolos, illetve Windows Forms alapú grafikus alkalmazásokon keresztül.

1. feladat

Készítsünk egy konzolos alkalmazást, amely beolvassa egy szöveges fájl tartalmát! Távolítsuk el a szöveg elejéről és végéről a nem betű karaktereket, majd készítsünk statisztikát az így keletkezett szavakról: írjuk ki a konzolra, melyik szó hányszor fordul elő, az előfordulások szerint csökkenő sorrendben.

A Visual Studio 2019 elindítása után válasszuk a `Create a new project` menüpontot.

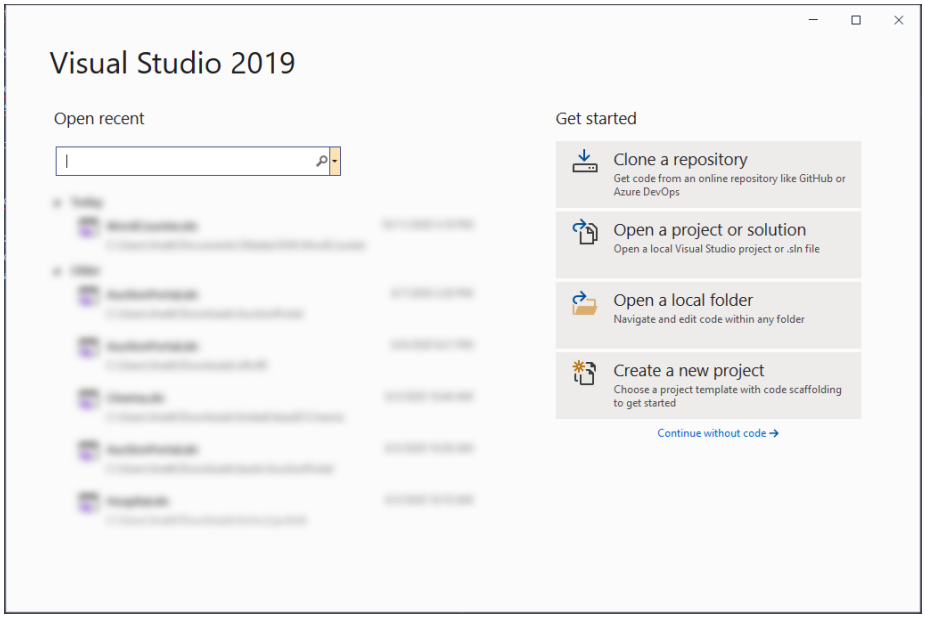

Figure 1: Új solution létrehozása

A következő ablakban válasszuk ki a **Console App (.NET Core)** projektsablont.

A Next gombra való kattintás után megadhatjuk a projekt és a solution nevét. A .NET megoldásokban (Solution) gondolkodik, egy Solution több projektet is tartalmazhat. Alapértelmezetten a solution neve megegyezik az első projekt nevével, de ezt lehet módosítani. Az induló projekt a Solution Explorerben a projekt nevén jobb klikk, majd a `Set as StartUp Project` menüpontot választva

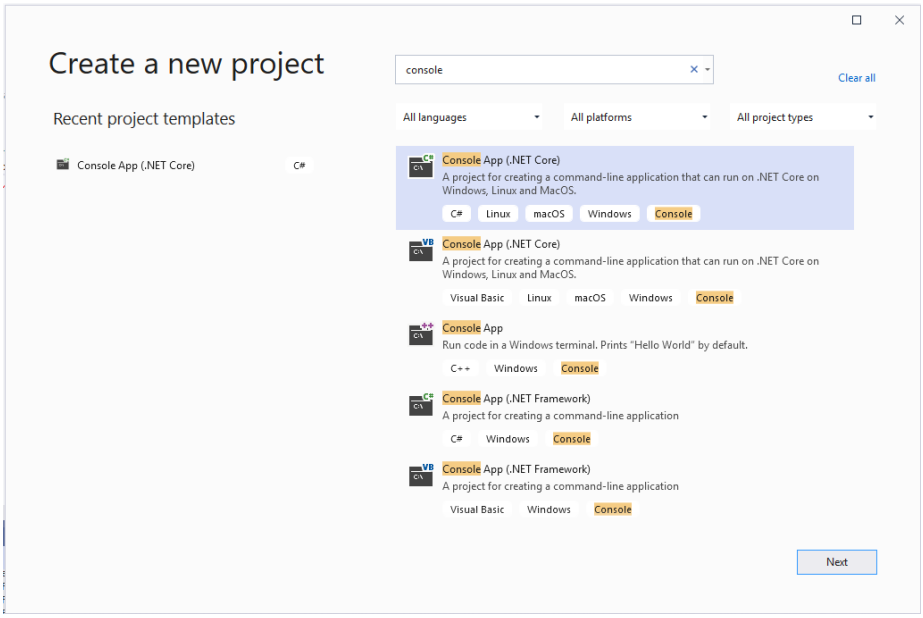

Figure 2: Új projekt létrehozása

adható meg. A sablon mindössze egy `Program` nevű osztályt generál, amelyben a statikus `Main` függvény található.

A fájlbeolvasáshoz és a számítások elvégzéséhez hozzunk létre egy új fájlt `Statistics.cs` néven. Ebben automatikusan létrejön egy osztály ugyanezen a néven. Készítsünk egy `FileContent` nevű tulajdonságot (*property*), amely legyen publikus, a típusa legyen `String`, és rendelkezzen egy publikus `get` és egy privát `set` accessorral ("elérő")!

Fájl tartalmának beolvasása

A beolvasás két lépésből áll. Első lépésben bekérünk a felhasználótól egy érvényes abszolút útvonalat egy szöveges (txt kiterjesztésű) fájlhoz. Ezt végezzük a `Main` függvényben, amely addig olvas be a konzolról, amíg egy létező szöveges fájlt nem kapunk. A beolvasást a `Console.Readline()` függvénnyel végezzük. A kiterjesztést a `System.IO.Path.GetExtension()` metódussal, a létezés ellenőrzését a `System.IO.File.Exists()` metódussal végezzük.

A második lépés a fájl tartalmának beolvasása. Ehhez készítsük el a `Load` nevű függvényt a `Statistics` osztályban, amely paraméterül a beolvasott útvonalat várja. A metódusban a `FileContent` értékét írjuk felül a `System.IO.File.ReadAllText(path)` hívással, amely stringként adja vissza a fájl tartalmát. A `ReadAllText()` `System.IO.IOException` kivételt okozhat, ezt azonban majd csak a hívó metódusok fogják elkapni.

Hívjuk meg a `Main`-ben a `Load` függvényt, és egy `try-catch` blokkban kezeljük a kivételt! A `catch` ágban írjuk ki a kivétel üzenetét (`Message`) a `Console.WriteLine()` paranccsal.

Szavak előfordulásának megszámlálása

A szövegen végzett műveletek elvégzéséhez definiáljunk egy `CountWords` nevű metódust a `Statistics` osztályban! A szavak előfordulásainak számát egy `String - int` párokat tartalmazó konténerben fogjuk tárolni (`Dictionary<String, int>`). Ezt a konténert is tulajdonságként vegyük fel, hasonlóan a `FileContent`-hez. Adjuk neki a `WordCount` nevet, és inicializáljuk az osztály konstruktorában.

A `CountWords`-ben a szöveget tördeljük szavakra a `Split()` függvény hívásával! Ha ezt paraméter nélkül hívjuk meg, automatikusan a *whitespace*-eknél fogja szétvágni a szöveget. A `Split()` eredménye egy string tömb (`words`). A tömbben ekkor még lehetnek üres stringek, ezeket egy lambdakifejezéssel távolítsuk el a tömbből:

```
words = words.Where(s => s.Length > 0).ToArray();
```

A `words` tartalmát tehát felülírjuk a nem üres stringekkel. A `Where` hívás egy `IEnumerable`-t ad vissza, amit tömbbé kell konvertálnunk az értékadáshoz (`ToArray()`).

A maradék szavakon iteráljunk végig. A szavak elejéről vegyük le az összes nem betű karaktert. A karaktereket a `Char.IsLetter()` metódussal vizsgáljuk meg! Az aktuális szóból a `Remove` metódus hívásával távolíthatjunk el karaktereket. Ennek paraméterként adjuk meg az eltávolítás kezdőindexét (ez a 0. indexű elem lesz), és az eltávolítandó karakterek számát. Ne felejtsük el mindig vizsgálni, hogy van-e még karakter a szóban, így kerüljük el a túlindexelést.

```
while (words[i].Length > 0 && !Char.IsLetter(words[i][0]))
{
    words[i] = words[i].Remove(0, 1);
}
```

Ugyanezt tegyük meg fordítva, a szó utolsó karakterétől kezdve. A `^1` indexeléssel hivatkozhatunk a szó utolsó karakterére (természetesen a szó hosszából számított indexelés is jó megoldás.)

```
while (words[i].Length > 0 && !Char.IsLetter(words[i][^1]))
{
    words[i] = words[i].Remove(words[i].Length - 1, 1);
}
```

Ezután lehetséges, hogy újabb üres string keletkezett (pl. egy évszám esetén). Ezt vizsgáljuk meg a `String.IsNullOrEmpty()` függvénnyel, és ha igazat kapunk, lépjünk tovább a következő szóra!

Az aktuális szót alakítsuk át a `ToLower()` függvénnyel, hogy csak kisbetűket tartalmazzon, így elkerüljük a kis- és nagybetűs különbségekből adódó duplikációkat.

Vizsgáljuk meg, hogy a szó benne van-e már a `WordCount` kulcsai között (`ContainsKey()`). Ha igen, növeljük a szóhoz tartozó értéket 1-gyel (használható az index operátor `[]`), egyébként vegyük fel a szót a `WordCount`-ba az `Add()` metódussal, értéke 1 legyen.

A `Main`-ben ezután hívjuk meg `CountWords` függvényt. A Dictionary egy rendezetlen típus, ezért a kiírás előtt rendeznünk kell az adatokat az előfordulások száma szerint. Rendezzük az értékek szerint `WordCount`-ot az `OrderByDescending()` metódussal, amelynek a rendezés feltételét egy lambdakifejezésként adjuk át, majd az eredmény `IEnumerable`-t mentsük el egy lokális változóba.

A rendezést LINQ lekérdezéssel, többféle szintaxissal is végezhetjük.

1) *Method Syntax:*

```
var pairs = statistics.WordCount.OrderByDescending(p => p.Value);
```

2) *Query Syntax:*

```
var pairs = from pair in WordCount
            orderby pair.Value descending
            select pair;
```

A két módszer ugyanazt az eredményt adja. Ezután a `pairs`-en végigiterálhatunk egy `foreach` ciklussal, hogy kiírjuk a kulcs-érték párokat.

```
foreach (var pair in pairs)
{
    Console.WriteLine("{0}: {1}", pair.Key, pair.Value);
}
```

2. feladat

Készítsünk az előző feladathoz grafikus felületet, amelyen megjelenítjük a betöltött fájl szövegét és a szavak előfordulásainak számát!

A Solution Explorerben a solution nevére jobb klikk után válasszuk ki az **Add –> New project...** menüpontot. Válasszuk a **Windows Forms (.NET Core)** sablont! A projektnek adjuk a *VisualizeWordCounter* nevet.

A generált kód egy `Form`-ból (`Form1`) és egy `Program` nevű osztályból áll, ebben található a `Main` függvény. A Form1 az alkalmazásunk elsődleges ablaka. A Form1-re kétszer kattintva előhozhatjuk a Designer nézetet, ahol manuálisan helyezhetünk fel vezérlőket az ablakra. A Form1.cs-t lenyitva láthatjuk a Designer mögött lévő generált kódot (`Form1.Designer.cs`), ebbe ne nyúljunk bele. A

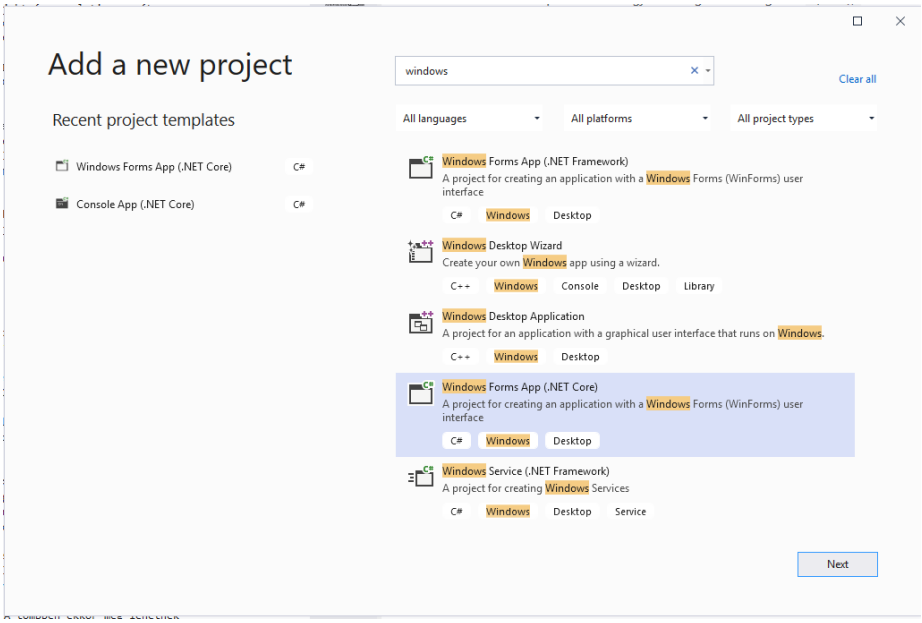

Figure 3: Windows Forms (.NET Core) projekt létrehozása

`Form1` szerkesztendő kódját a `Form1.cs`-re jobb klikkelve, a *View code* menüponttal hozhatjuk elő. A továbbiakban ebben a fájlban és a Designer nézetben fogunk dolgozni. Nevezzük át a Form1.cs fájlt `WordCountDialog`-ra!

A Designerben helyezzük fel a következő vezérlőket az ablakra:

- `MenuStrip`: az ablak tetején, a címsor alatt helyezkedjen el, ez lesz az alkalmazás menüsávja. A vezérlőre jobb klikkelve válasszuk ki az *Edit items...* menüpontot. Az *Add* gombbal adjunk új menüelemet a menüsávhoz.
  - A jobb oldalon látható ablakban be tudjuk állítani a menüelem nevét (`(Name)`), ez legyen `fileMenu`, és szövegét (`Text`), ez legyen `File`. A `fileMenu`-t kijelölve keressük meg a tulajdonságok között a `DropDownItems` elemet, és a `(Collection)`-re kattintva válasszuk ki a mellette megjelenő három pontot. Ezzel egy hasonló menüt kapunk, ahol a `fileMenu`-hoz adhatunk új menüelemeket az *Add* gombbal.
    * Adjunk a `fileMenu`-hoz egy új elemet, ennek neve legyen `openFileDialogMenuItem`, felirata pedig legyen `Open file dialog`.
    * Egy második menüelem neve legyen `countWordsFileDialog`, a felirata pedig legyen `Count words`.
- `TextBox`: ebben fogjuk megjeleníteni a szöveges fájl tartalmát. Helyezzük az ablak bal oldalára, és méretezzük, hogy kb. az ablak felét foglalja el. A Solution Explorer alatt megjelenik a Designerben aktuálisan kijelölt

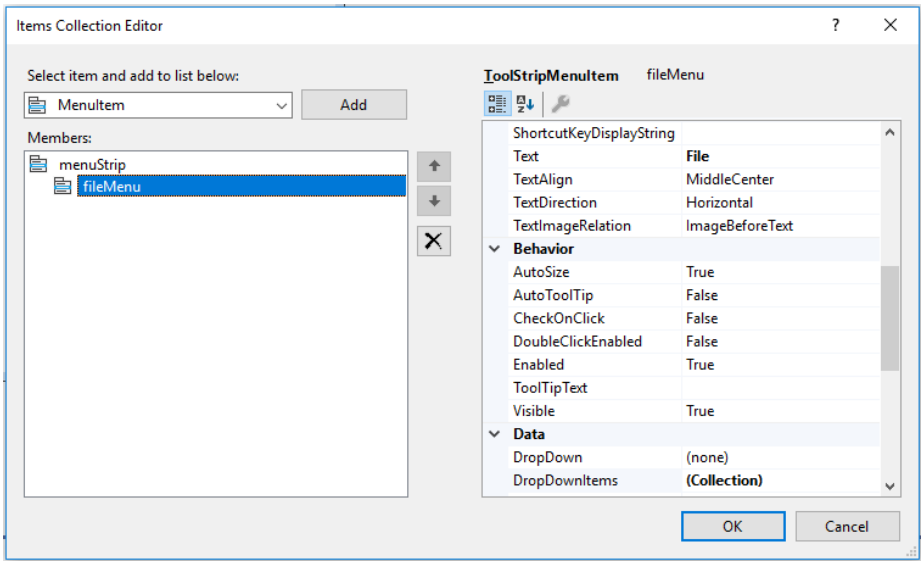

Figure 4: Menüpont hozzáadása a menüsávhoz.

elemhez tartozó tulajdonságok listája (`Properties`), amelyben módosítani tudjuk a vezérlő viselkedését, eseménykezelőket adhatunk az eseményeihez stb. A `TextBox (Name)` tulajdonságát írjuk át `textBox`-ra, majd a `ScrollBars` tulajdonságot állítsuk `Vertical`-ra, hogy görgethető legyen a mező. A `ReadOnly` tulajdonságot állítsuk igazra, hogy a felületen keresztül ne legyen szerkeszthető a szöveg.

- `ListBox`: ebben jelenítjük meg a szavakat és előfordulásaik számát. Helyezzük az ablak jobb oldalára, és méretezzük át, hogy kitöltse a maradék helyet. A `Properties` ablakban adjuk neki a `listBoxCounter` nevet.

A teljes ablak kijelölésével láthatjuk a `Properties` ablakban az ablak tulajdonságait. Legyen a fejléc szövege (`Text`) `Word counter`! A `FormBorderStyle` tulajdonságot állítsuk `FixedSingle`-re, a `MaximizeBox` tulajdonságot pedig hamisra, így az ablak fix méretű lesz.

Mivel fel fogjuk használni az előző feladat `Statistics` osztályát, el kell érnünk a másik projekt névterét. Jobb klikkeljünk a Solution Explorerben a `VisualizeWordCounter` projektre, majd válasszuk ki az *Add –> Project Reference...* menüpontot. Pipáljuk be a `WordCounter` projektet, majd okézzuk le az ablakot.

Nyissuk meg a `WordCountDialog.cs` fájlt (ami semmképpen nem a `WordCountDialog.Designer.cs`)! Ebben a fájlban írjuk meg a menüelemekhez tartozó eseménykezelőket, amelyek a fájlbeolvasást és a szavak számlálását fogják végezni, és kitöltik a `textBox`-ot és a `listBoxCounter`-t. A

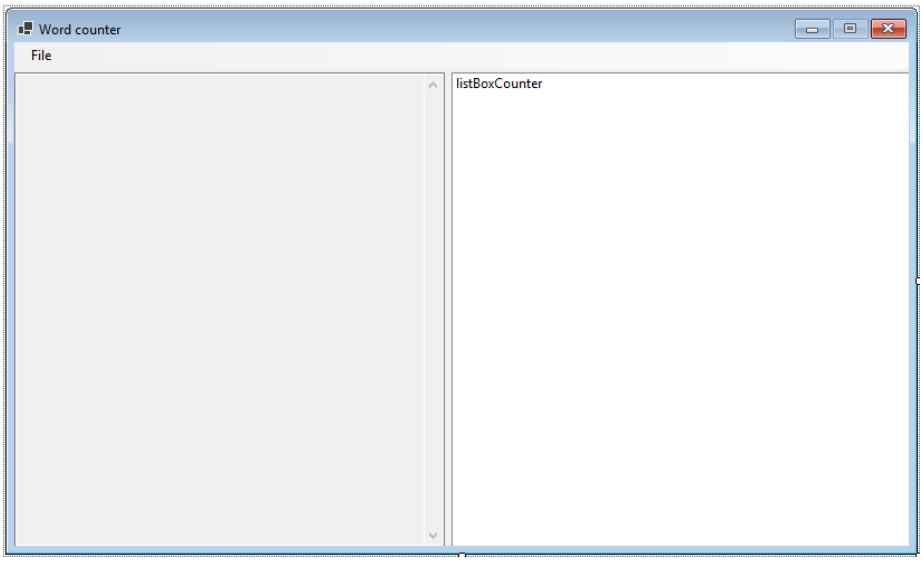

Figure 5: Az alkalmazáshoz tartozó ablak nézete a Designerben.

usingok közé vegyük fel a `WordCounter` névteret!

Vegyünk fel egy privát `Statistics` típusú adattagot (`statistics`), és inicializáljuk a konstruktorban. Definiáljuk az `openFileDialogMenuItem`-hez tartozó eseménykezelőt! Az eseménykezelő egy privát, `void` visszatérési típus metódus, amely a küldő objektumot és egy eseményargumentumot vár paraméterül. A neve legyen `OpenDialog`.

```
private void OpenDialog(object sender, EventArgs e)
```

A fájlt most egy fájltallózó dialóguson keresztül fogjuk megnyitni. Használjuk ehhez az `OpenFileDialog` osztályt. A dialógus kezdeti könyvtárát az `InitialDirectory`, a megengedett fájltípusokat a `Filter` tulajdonságon keresztül állíthatjuk be. Ha azt szeretnénk, hogy minden megnyitáskor a kezdeti könyvtár jelenjen meg, állítsuk a dialógus `RestoreDirectory` tulajdonságát igazra.

A dialógus `ShowDialog()` metódusát meghívva megjelenítjük a fájltallózót. Ha itt sikeresen kiválasztottunk egy fájlt, hívjuk meg a `statistics` `Load()` metódusát. Paraméterként az `openFileDialog` `FileName` propertyjét adjuk át. Ne feledjük, hogy itt el kell kapnunk az esetleges kivételt. A `catch` ágban jelenítsünk meg egy `MessageBox`-ot a kivétel szövegével (`Message`).

Egy `StreamReader` objektum (`reader`) használatával olvassuk be a `fileStream` tartalmát! A `StreamReader` `ReadToEnd` metódusa egy stringbe beolvassa a stream teljes tartalmát, ezt adjuk értékül a `FileContent`-nek. Ez a művelet `System.IO.IOException` kivételt válthat ki, ezt kezeljük `try-catch` blokkban.

Kivétel esetén jelenítsünk meg egy `MessageBox`-ot a kivétel szövegével.

Megspórolhatjuk a szöveg ismételt betöltését abban az esetben, ha ugyanazt a fájlt ugyanazzal a tartalommal próbáljuk betölteni, mint ami már a `textBox`-ban van. Vizsgáljuk meg, hogy a `statistics.FileContent` egyezik-e a `textBox` tartalmával (`Text`), feltéve, hogy előbbi nem üres! Ha a betöltendő szöveg egyezik a korábbival, térjünk vissza.

HA `textBox` `Text` tulajdonságának adjuk értékül a `FilContent` tartalmát, így megjelenik a `textBox`-ban a fájlból beolvasott szöveg. A `listBoxCounter` `Items` kollekciójának tartalmát töröljük a `Clear()` metódussal.

```
private void OpenDialog(object sender, EventArgs e)
{
   using (OpenFileDialog openFileDialog = new OpenFileDialog())
   {
      openFileDialog.InitialDirectory = "C:\\";
      openFileDialog.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
      openFileDialog.RestoreDirectory = true;

      if (openFileDialog.ShowDialog() == DialogResult.OK)
      {
         FilePath = openFileDialog.FileName;
         var fileStream = openFileDialog.OpenFile();

         try
         {
            statistics.Load(openFileDialog.FileName);
         }
         catch (System.IO.IOException ex)
         {
            MessageBox.Show("File reading is unsuccessful!\n" + ex.Message,
            "Error", MessageBoxButtons.OK);
         }

         if (!String.IsNullOrEmpty(statistics.FileContent)
             && statistics.FileContent == textBox.Text)
              return;

         listBoxCounter.Items.Clear();
         textBox.Text = FileContent;
      }
   }
}
```

Az `OpenDialog`-ot kössük össze a konstruktorban az `openFileDialogMenuItem` `Click` eseményével.

```
openFileDialogMenuItem.Click += OpenDialog;
```

Eseménykezelőt a `-=` operátorral vehetünk le egy eseményről.

Definiáljunk egy új eseménykezelőt `CalculateStatistics` néven! Láthatósága, visszatérési típusa, paraméterei ugyanazok legyenek, mint az előző eseménykezelőnek.

Ha még nem olvastunk be fájlt, azt szeretnénk, ha ez a menüpont feldobna egy üzenetet. Ha a `statistics.FileContent` üres (amit a `String.IsNullOrEmpty()` metódussal tudunk megvizsgálni), dobjunk fel egy `MessageBox`-ot, amely tájékoztat, hogy még nincs beolvasott szöveg, majd térjünk vissza.

Hívjuk meg a `statistics` `CountWords()` metódusát, majd az előző feladathoz hasonlóan rendezzük a `WordCount` elemeit.

A `listBoxCounter` `BeginUpdate()` metódusát meghívva kezdhetjük a párok kiírását. A `Clear()` metódussal itt is töröljük a `listBoxCounter` `Items` kollekcióját. A rendezett párokon végigiterálva adjuk hozzá az `Items`-hez szövegként a párok kulcsát és értékét, majd az `EndUpdate()` metódussal zárjuk le a `listBoxCounter` szerkesztését.

```
listBoxCounter.BeginUpdate();
foreach (var pair in pairs)
{
    listBoxCounter.Items.Add(pair.Key + ": " + pair.Value);
}
listBoxCounter.EndUpdate();
```

A `CalculateStatistics` eseménykezelőt az előzőhöz hasonlóan kössük rá a `countWordsMenuItem` `Click` eseményére.