


Szövegfüggvények ABC-je

FONTOS!

A függvényeket össze is lehet fűzni! Pl.

```
s.Replace(„a”, „cica”).Contains(„kutya”);
```


Ehhez mindig tudni kell, hogy az adott függvénynek mi a visszatérési értéke. A visszatérési értéket innen lehet tudni:



```
string string.Replace(char oldChar, char newChar) (+ 1 overload(s))  
Returns a new string in which all occurrences of a specified Unicode character in this instance are replaced with another specified Unicode character.
```

Ez alapján a Replace visszatérési értéke egy string típus, ami azt jelenti, hogy szöveget ad vissza. A leírásából az is kiolvasható, hogy azt a szöveget fogja visszaadni, amiben már kicserélte a kért adatokat (az „a” betűt „cica”-ra), így ezzel tudunk tovább dolgozni. Ez ugyan az, mintha kimentenénk külön változóba, majd arra hívnánk meg a következő függvényt. Mind a két megoldás egyformán jó.

A Contains visszatérési értéke egy logikai érték lesz, tehát igaz / hamis.



```
bool string.Contains(string value) (+ 2 overload(s))  
Returns a value indicating whether the specified System.String object occurs within this string.  
  
Exceptions:  
System.ArgumentNullException
```

Ez a függvény nem fogja visszaadni a készített szöveget, mivel ez egy eldöntést végez (tartalmazza-e) és annak az eredményét adja vissza.

FONTOS2!

A szövegfüggvények NEM módosítják az alap szöveget! Pl. amennyiben van egy string s = „szöveg”; változónk, ha végrehajtjuk az alábbi parancsot:

```
s.Replace(„ö”, „x”);
```

Akkor az s változóban az alábbi szöveg lesz: „szöveg”

Ennek oka, hogy a változás csak akkor kerül mentésre ha visszaadjuk az értéket a változónak, tehát:

```
s = s.Replace(„ö”, „x”);
```

Minden egyéb esetben megcsinálja amit kértünk, de NEM menti el. Ez akkor jó, ha nincs vele további dolgunk, pl amennyiben csak annyi a feladat, hogy cseréld ki a betűt és utána NEM használd tovább ezt a szöveget, akkor megoldás lehet ez:

```
Console.WriteLine(s.Replace(„ö”, „x”)); -> így kiírtad a konzolablakba de nem mentetted el.
```

CompareTo

A CompareTo metódust két szöveg összehasonlítására használjuk, eredménye -1, 0, és 1 lehet (kisebb, egyenlő vagy nagyobb). Rendezésnél hasznos.

Két mintaszövegünk: string a = „Kis Pista”; és string b = „Kis Pál”;

Amennyiben ezt a két szöveget összehasonlítjuk: `a.CompareTo(b)`; úgy 1 lesz az eredményünk, azaz az „a” változó nagyobb, tehát ABC sorrend szerint a „b” változó értéke kerülne előrébb.

Contains

Megvizsgálja, hogy a szövegünk tartalmaz-e egy adott karaktert, szöveget.

Pl. arra vagyunk kíváncsiak, hogy tartalmaz-e a szövegünk „a” betűt:

Mintaszöveg: `string s = „kiscica”;`

`s.Contains('a');` -> true

`s.Contains('x');` -> false

`s.Contains(„a”);` -> true

`s.Contains(„cica”);` -> true

Count, Length

A szöveg hosszát adják vissza.

Használata: `s.Count()`; VAGY `s.Length`;

EndsWith

Visszaadja, hogy az adott szöveg a keresettre végződik-e.

Mintaszövegek: `string s1 = „Kis cica”; string s2 = „Nagy kutya”;`

`s1.EndsWith(„cica”)` -> igaz

`s2.EndsWith(„cica”)` -> hamis

`s1.EndsWith(„Cica”)` -> hamis (kis- és nagybetűre érzékeny)

StartsWith

Az [EndsWith](#)-hez hasonló csak azzal ellentétben ez a szöveg elejét vizsgálja.

Mintaszövegek: `string s1 = „Kis cica”; string s2 = „Kis”;`

`s1.StartsWith(s2)` -> igaz

`s1.StartsWith(„Kis”)` -> igaz

`s2.StartsWith(s1)` -> hamis

IndexOf

Kétféle értéke lehet: -1 amennyiben nem található a keresett szöveg vagy -1-nél nagyobb szám, amennyiben található, ebben az esetben az első találat indexét adja vissza.

Mintaszöveg: `string s = „Tigris”;`

`s.IndexOf(„T”);` -> 0

`s.IndexOf(„t”);` -> -1

`s.IndexOf(„i”); -> 1`

LastIndexOf

Az [IndexOf](#)-hoz hasonló csak az első helyett az utolsó előfordulást keresi. A visszatérési értékei ugyan azok lehetnek.

Mintaszöveg: `string s = „Tigris”;`

`s.LastIndexOf(„T”); -> 0`

`s.LastIndexOf(„t”); -> -1`

`s.LastIndexOf(„i”); -> 4`

Insert

A listánál tanult Insert-el egyenértékű, meg lehet adni neki hogy hanyadik pozícióra és mit szúrjon be. Amennyiben pl. a 3. pozícióra szúrunk be, úgy ami eddig a 3. helyen állt az mostantól a beszúrt szövegünk utánra kerül és így tolódik minden.

Pl. `string s = „Cica”;`

`s.Insert(1, „meow”); -> Cmeowica`

Remove

A szöveg bizonyos részeinek eltávolítására használható. Kétféleképpen lehet paraméterezni, egyik esetben az adott indextől az összes karaktert törli (1. példa), másik esetben az adott indextől adott darabszámú karaktert töröl (2. példa)

Mintaszöveg: `string s = „szövegem”;`

`s.Remove(3); -> szö`

`s.Remove(3, 3); -> szöem`

Replace

Egy szövegben egy rész kicserélése egy másikra. Használható karakterek és szövegek cseréjére is, illetve ezt szokás használni arra is, ha el szeretnénk „távolítani” egy karaktert vagy karaktersorozatot.

Mintaszöveg: `string s = „Az oroszlán aludni megy”;`

`s.Replace(‘a’, ‘ü’); -> Az oroszlán üludni megy`

`s.Replace(„aludni”, „pihenni”); -> Az oroszlán pihenni megy`

`s.Replace(„a”, „kis”); -> Az oroszlán kisludni megy`

`s.Replace(„aludni”, „”); -> Az oroszlán megy`

Split

Szöveg darabolása a megadott karakter mentén. Leggyakoribb példa, hogy kérj be egy nevet, majd tárold el külön a vezeték- és keresztnévét (feltéve, hogy nincs több keresztnéve).

Mintaszöveg: `string s = „Veznevem Kernevem”;`

s.Split(' '); -> visszaad egy két elemű tömböt, aminek a 0- helyén a „Veznevem”, 1. helyén pedig a „Kernevem” szöveg áll.

Substring

Egy szövegrész kivágása, reverze a [Remove](#)-nak (tehát itt nem azt adod meg, hogy mit töröljön, hanem hogy mi maradjon..). Kétféleképpen paraméterezhető, vagy a kezdőindexet adjuk csak meg, ez esetben attól az indextől mindent kivág és megtart (a többit eldobja) (1. példa), vagy megadjuk a kezdőindexet és onnantól a darabszámot (2. példa).

Mintaszöveg: string s = „Panda”;

s.Substring(2); -> „nda”;

s.Substring(1, 2); -> „an”;

Lehet azt is, hogy csak az első és az utolsó karaktert vágja le:

s.Substring(1, s.Length – 2); -> and

ToUpper

Csupa nagybetűssé alakítja a szöveget

Mintaszöveg: string s = „aSd”;

s.ToUpper(); -> ASD

ToLower

Csupa kisbetűssé alakítja a szöveget

Mintaszöveg: string s = „aSd”;

s.ToLower(); -> asd

Trim

Paraméterezés nélkül arra használható, hogy eltávolítsa a vezető és végző üres szóközöket.

Mintaszöveg: string s = „ sok szóköz ”;

s.Trim(); -> „sok szóköz”

FONTOS, hogy a két szó közti fölösleges szóközöket **NEM** törölte! Csak az elejéről és a végéről!

TrimEnd

Ugyan az, mint a [Trim](#), csak kizárólag a végéről töröl.

TrimStart

Ugyan az, mint a [Trim](#), csak kizárólag az elejéről töröl.

Distinct*

Visszaadja az összes előforduló karaktert egyszer. Visszatérési értéke egy tömb.

Mintaszöveg: string s = „Az alma barna”;

s.Distinct().ToList(); -> visszaad egy 9 elemű listát

Ezt fel lehet így is használni, vagy ki is lehet akár írni a szöveget a felhasználónak úgy, hogy minden betűt csak 1x lát:

```
List<char> karakterek = s.Distinct().ToList();  
for (int i = 0; i < karakterek.Count; i++)  
{  
    Console.Write(karakterek [i]);  
}
```