Szövegkezelés

C# egy erőteljes és sokoldalú programozási nyelv, amely különösen jól használható szövegkezelésre a beépített funkcióinak és osztályainak köszönhetően.

A C# szövegkezelési képességei rendkívül gazdagok, és számos különféle műveletet hajthatunk végre a string típusú változókkal. Az alábbiakban részletesebben bemutatom a leggyakoribb sztringműveleteket:

1. Sztring összefűzés

Sztringeket összefűzhetünk különböző módszerekkel:

- **+ Operátor:** Egyszerű és közvetlen módja a sztringek összefűzésének.

```
string firstName = "John";
string lastName = "Doe";
string fullName = firstName + " " + lastName; // "John Doe"
```

- **String.Concat() Metódus:** Több sztring összefűzésére is használható.

```
string fullName = String.Concat(firstName, " ", lastName);
```

- **String Interpolation:** Könnyen olvasható forma, amely változókat és kifejezéseket illeszt be a sztringbe.

```
string fullName = $"{firstName} {lastName}";
```

2. Substring

A Substring metódus segítségével egy szöveg részét tudjuk kivágni.

- **Substring(int startIndex):** Egy adott indexről kezdve adja vissza a szöveget.

```
string text = "Hello, World!";
string sub = text.Substring(7); // "World!"
```

- **Substring(int startIndex, int length):** Kivág egy meghatározott hosszúságú szövegrészt.

```
string sub2 = text.Substring(0, 5); // "Hello"
```

3. Karakterek kicserélése és eltávolítása

- **Replace:** Egy adott karakter vagy sztring cseréje másikra.

```
string sentence = "I love apples.";
string newSentence = sentence.Replace("apples", "oranges"); // "I love oranges."
```

- **Remove:** Karakterek eltávolítása egy adott helyről.

```
string text = "Hello, World!";
string removed = text.Remove(5, 7); // "Hello"
```

4. Szöveg hosszának lekérdezése

A Length tulajdonság segítségével lekérdezhetjük a sztring hosszát.

```
string text = "Hello";
int length = text.Length; // 5
```

5. Szöveg helyettesítése és formázása

- **Trim, TrimStart, TrimEnd:** Szóközök eltávolítása.

```
string text = "  Hello  ";
string trimmed = text.Trim(); // "Hello"
```

- **ToUpper és ToLower:** Szöveg átalakítása nagy- vagy kisbetűssé.

```
string upper = "hello".ToUpper(); // "HELLO"
string lower = "WORLD".ToLower(); // "world"
```

6. Index lekérdezése

A IndexOf metódus segítségével megtudhatjuk, hogy egy adott karakter vagy szöveg hol található a sztringben.

```
string text = "Hello, World!";
int index = text.IndexOf('W'); // 7
```

7. Adott karakter darabszámának meghatározása

A Count metódust a System.Linq névtérrel együtt használhatjuk, hogy megszámoljuk, hányszor fordul elő egy karakter a sztringben.

```
using System.Linq;

string text = "Hello, World!";
int count = text.Count(c => c == 'o'); // 2
```

for cilkussal!!!! Itt nincs olyan mint pythonban.

8. Tartalmaz-e?

A Contains metódus segítségével ellenőrizhetjük, hogy egy sztring tartalmaz-e egy adott karaktert vagy szöveget.

```
string text = "Hello, World!";
bool containsHello = text.Contains("Hello"); // true
bool containsX = text.Contains('x'); // false
```

9. Feldarabolás karakter mentén

A Split metódus segítségével egy szöveget több részre darabolhatunk egy megadott karakter vagy szöveg mentén.

```
string text = "apple,banana,cherry";
string[] fruits = text.Split(','); // {"apple", "banana", "cherry"}
```

10. Karakter kódja és kódból karakter

- **Karakter Kódja:** A char típusú karakterek numerikus kódját a Convert.ToInt32() metódussal kérdezhetjük le.

```
char letter = 'A';
int code = Convert.ToInt32(letter); // 65
```

- **Kódból Karakter:** Egy numerikus kódot karakterré alakíthatunk a Convert.ToChar() metódussal.

```
int code = 65;
char letter = Convert.ToChar(code); // 'A'
```

11. Sztringek összehasonlítása

A string típus különböző metódusokat kínál az összehasonlításra:

- **Equals:** Két sztring egyenlőségének ellenőrzése.

```
bool isEqual = "hello".Equals("hello"); // true
```

- **CompareTo:** Sztringek lexikográfiai összehasonlítása.

```
int comparison = "apple".CompareTo("banana"); // -1, mert "apple" előbb van mint "banana"
```

12. StringBuilder használata

A StringBuilder hatékony megoldás, ha sok módosítást kell végezni a szövegen, mivel nem hoz létre új sztring objektumot minden módosításnál.

```
using System.Text;

StringBuilder sb = new StringBuilder();
sb.Append("Hello");
sb.Append(", ");
sb.Append("World!");

string result = sb.ToString(); // "Hello, World!"
```