第11章 string型とchar型　その4

エスケープシーケンスを用いると「改行」とか「タブ」など通常の文字では表せない文字を表現できます。

エスケープシーケンスは、C/C++のそれとほとんど同じです。

エスケープシーケンス	意味
¥r	キャリッジリターン
¥n	改行
¥t	タブ
¥v	垂直タブ
¥'	シングルクォーテーション
¥"	ダブルクォーテーション
¥b	バックスペース
¥¥	円記号(¥)
¥f	改ページ
¥0	ヌル文字

改行のみを出力するとき

```
Console.WriteLine();
```

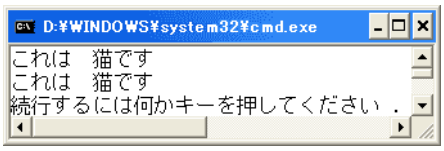
としても

```
Console.Write("¥n");
```

としても同じことになります。

```
// escape01.cs

using System;

class escape01
{
    public static void Main()
    {
        string str;
        string[] strarray = new string[] { "これは", "猫です" };
        string stra = "これは¥t猫です¥n";

        str = string.Join("¥t", strarray);
        str = string.Concat(str, "¥n");

        Console.Write(str);
        Console.Write(stra);
    }
}
```

実行結果は、次のようになります。



stringクラスには、Replaceメソッドというものがあります。これは、文字や文字列を置き換えます。

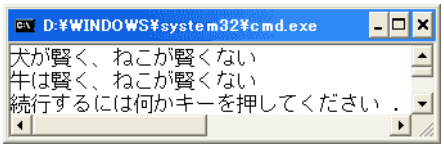
```
public string Replace (
    char oldChar,
    char newChar
)
public string Replace (
    string oldValue,
    string newValue
)
```

サンプルを見てみましょう。

```
// replace01.cs
```

```
using System;

class replace01
{
    public static void Main()
    {
        string str = "犬は賢く、ねこは賢くない";

        str = str.Replace('は', 'が');
        Console.WriteLine(str);

        str = str.Replace("犬が", "牛は");
        Console.WriteLine(str);
    }
}
```

実行結果は次のようになります。



最初の置き換えでは、文字'は'を'が'に置き換えています。

次の置き換えでは、文字列"犬が"を"牛は"に置き換えています。

さて、stringクラスにはもっと便利なメソッドがあります。Splitメソッドは 文字列を指定の区切り文字で区切って文字列配列にすることができます。C/C++のstrtok関数と似ています。いくつかのバージョンがあります。

```
public string[] Split (
    params char[] separator
)
```

separator区切り文字配列で、このオブジェクトを区切って文字列配列に格納します。区切り文字は文字列配列に含まれません。

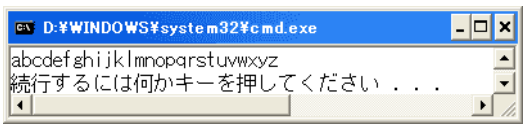
```
// split01.cs

using System;

class split01
{
    public static void Main()
    {
        string str = "abc,def,gh¥nijk¥tlmn,opq¥trst,uvw,xyz";

        char[] sep = new char[] { ',', '¥n', '¥t' };

        string[] newstr = str.Split(sep);

        str = string.Concat(newstr);
        Console.WriteLine(str);
    }
}
```

区切り文字を','と'¥n'と'¥t'にして、配列sepを作っています。

そして、strを区切り文字で切って、部分文字列をnewstr配列に格納しています。newstr[0]には"abc", newstr[1]には"def",...が格納されます。

部分文字列の配列をConcatメソッドで結合してstrに格納しています。

結局strは、元の文字列から区切り文字を取り除いた文字列となります。



Splitメソッドには、取得する部分文字列配列の要素数をしてするバージョンもあります。

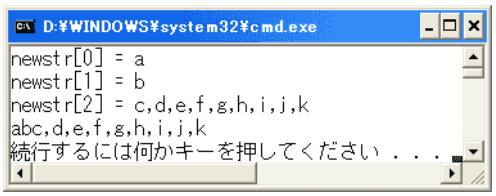
```
public string[] Split (
    char[] separator,
    int count
)
```

countに、取得する文字列配列の要素数を指定します。

```
// split02.cs

using System;
```

```
class split02
{
    public static void Main()
    {
        string str = "a,b,c,d,e,f,g,h,i,j,k";

        char[] sep = new char[] { ',' };
        string[] newstr = str.Split(sep, 3);

        Console.WriteLine("newstr[0] = {0}", newstr[0]);
        Console.WriteLine("newstr[1] = {0}", newstr[1]);
        Console.WriteLine("newstr[2] = {0}", newstr[2]);

        str = string.Concat(newstr);

        Console.WriteLine(str);
    }
}
```

この例では、strを','で区切って部分文字列を取得していますが、その要素数を3に限定しています。



区切り文字が2つ以上続くと、部分文字列に空の文字列が作られてしまいます。これを防ぐオプション付きのバージョンもあります。

```
public string[] Split (
    string[] separator,
    StringSplitOptions options
)
```

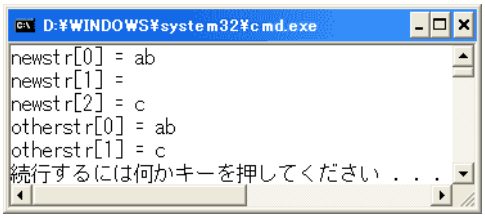
optionsは、StringSplitOptions列挙体でメンバは次の2つのみです。列挙体については後の章で解説します。

メンバ	意味
None	戻り値には空の文字列を含むものも含まれる
RemoveEmptyEntries	空の文字列を含むものは含まれない

簡単なサンプルを見てみましょう。

```
// split03.cs

using System;

class split03
{
    public static void Main()
    {
        string str = "ab,,c";
        char[] sep = new char[] { ',' };

        string[] newstr = str.Split(sep, StringSplitOptions.None);
        Console.WriteLine("newstr[0] = {0}", newstr[0]);
        Console.WriteLine("newstr[1] = {0}", newstr[1]);
        Console.WriteLine("newstr[2] = {0}", newstr[2]);

        string[] otherstr = str.Split(sep, StringSplitOptions.RemoveEmptyEntries);
        Console.WriteLine("otherstr[0] = {0}", otherstr[0]);
        Console.WriteLine("otherstr[1] = {0}", otherstr[1]);
    }
}
```

実行結果は次のようになります。



newstr配列には、空の文字列が含まれていますが、otherstr配列には含まれていません。

他にもいくつかのオーバーロードバージョンがありますが、MSDN等で調べてみてください。

Update 14/Aug/2006 By Y.Kumei