



## Yazı Biçimlendirme

05.09.2006

Bildiğiniz gibi C# dilinde decimal tipte bir m değışkeni řu řekilde yazıya dönüřtürölür.

```
using System;
class YazıBiçimlendirme
{
    static void Main()
    {
        double d=23.45;
        string s = d.ToString();
        Console.WriteLine(s);
    }
}
```

ToString metodunun kullanımı burada görüldüğünden biraz fazla esnektir. .net Framework, sayısal türleri yazıya dönüřtürürken istediğimiz biçimde biçimlendirip gösterme imkânı sunar. Meselâ:

```
using System;
class YazıBiçimlendirme
{
    static void Main()
    {
        double d=23.45;
        //string s = d.ToString();
        string s = d.ToString("G");
        Console.WriteLine(s);
    }
}
```

Buradaki G'nin anlamı (General) Genel formatlamadır. Varsayılan değer zaten G olduğı için yazmayabilirsiniz. Bu durumda yukarıdaki her iki örneğin de yapığı iş aynıdır. Farklı bir formatlama için tablodaki harfleri kullanabilirsiniz. Tablodaki harfleri büyük ya da küçük harfle yazmak fark etmiyor.

Harf	Anlamı	Açıklama
C	Currency	Para birimleri ile birlikte kullanılır. Üçer basamak ayırır.
D	Decimal	Sadece integer değerler için.
E	Exponential	Bilimsel gösterim için kullanılır.
F	Fixed point	Sayının bilimsel olmayan gösterimde gösterilmesini sağlar.
G	General	Üstel ve noktalı değerlerin çok küçük olduğı yerlerde.
N	Number	F ile aynıdır. Üçer basamak ayırır.
P	Percent	Yüzde ifadeleri için kullanılır. Sayı 100'e bölünür, üçer basamak ayırır ve % işareti gösterilir.
R	Roundtrip	Kayar nokta türler için kullanılır. Parse yöntemi ile geri dönüřtürülebilecek bir string oluşturur.
X	Hexadecimal	Tamsayı değerleri hexadecimal olarak gösterir.

Burada dokuz tane formatlama biçimi var. Bir tanesi için örnek verelim. m 0,123 değerini tutuyorsa aşağıdaki programın çıktısı %12,30 olacaktır.

```
using System;
class YazıBiçimlendirme
{
    static void Main()
    {
        double d = 0.123;
        string s = d.ToString("P");
        Console.WriteLine(s);
    }
}
```

C formatlama özelliği para sembolü göstermek için kullanılır. Bu sembol mutlaka dolar işareti olmak zorunda değildir. Bölge ve dil seçeneklerine göre değişir. Ama biz özellikle bir para birimin göstermek istiyorsak, meselâ dolar işareti göstermek istiyorsak aşağıdaki gibi yazmalıyız.

```
m.ToString("N") + "$"
```

İsterseniz harften sonra rakamda kullanabilirsiniz. Örneğin;

```
m.ToString("P5")
```

Buradaki rakamın anlamı yanındaki harfe göre değişiklik gösterebilir. Aşağıdaki tabloda ToString metodu ile sayıların nasıl kullanıldığı gösteriliyor.

Harf	Anlamı	Yazılacak Sayının Sonuca Etkisi
C	Currency	Virgülden sonra gösterilecek basamak sayısı (Varsayılan değer 2)
D	Decimal	Gösterilecek basamak sayısı (Eğer eksik olursa başına 0 eklenir.)
E	Exponential	Virgülden sonra gösterilecek basamak sayısı (Varsayılan değer 6)
F	Fixed point	Virgülden sonra gösterilecek basamak sayısı (Varsayılan değer 2)
G	General	Gösterilecek basamak sayısı.
N	Number	Virgülden sonra gösterilecek basamak sayısı (Varsayılan değer 2)
P	Percent	Virgülden sonra gösterilecek basamak sayısı (Varsayılan değer 2)
R	Roundtrip	Kullanılmaz
X	Hexadecimal	Gösterilecek basamak sayısı (Eğer eksik olursa başına 0 eklenir.)

Bir örnekle decimal tipteki bir değişkenin bütün formatlama çeşitlerini inceleyelim.

```
using System;
class NümerikFormatlama
{
    static void Main()
    {
        decimal m = 12345.345m;

        Console.WriteLine("Para formatında: " + m.ToString("C2"));
        Console.WriteLine("Üstel formatında: " + m.ToString("E2"));
        Console.WriteLine("Sabit noktalı formatında: " + m.ToString("F2"));
        Console.WriteLine("Genel formatında: " + m.ToString("G2"));
    }
}
```

```
        Console.WriteLine("Sayı formatında: " + m.ToString("N2"));
        Console.WriteLine("Yüzde formatında: " + m.ToString("P2"));
    }
}
```

ToString yöntemi bölge ve dil seçeneklerine göre farklı sonuçlar verebilen esnek parametrelere sahiptir. Programınız hiçbir değişiklik yapmadan seçili bölge ve dil seçenekleri doğrultusunda uygun sonuçlar üretebilmektedir. Türkiye için seçili ayarlar doğrultusunda programın sonucu aşağıdaki gibi olacaktır.

Yukarıda görülen ekran çıktısı incelenirse yuvarlama işlemleri var. Ama bu yuvarlama işlemleri Round yönteminin kurallarından daha farklı yapılmıştır. İki sayı arasındaki bir değer yuvarlanacaksa yukarı yuvarlanmıştır. 12345.345 sayısı 12345.34 olarak yuvarlanması gerekirken yukarı yuvarlanmış ve 12345.35 olarak yuvarlanmıştır. Eğer bu sizin işinizi görmüyorsa ToString işleminden önce Round ile yuvarlayınız.

Aşağıdaki örnekte 12345 tamsayı sayısı hexadecimal sayıya dönüştürülmektedir.

```
using System;

class NumerikFormatlama
{
    static void Main()
    {
        int i = 12345;
        string s = i.ToString("X8");
        Console.WriteLine(s);
    }
}
```

Burada 00003039 sayısı üretilir ve kısaltma sözkonusu değildir, 8 basamağa tamamlamak için başına dört tane 0 eklenmiştir. Ama aşağıdaki örnekte virgülden sonrası için 2 basamak belirtildiği için duyarlılık kaybı olacaktır.

```
using System;

class NumerikFormatlama
{
    static void Main()
    {
        int i = 12345;
        string s = i.ToString("E2");
        Console.WriteLine(s);
    }
}
```

Ekranında Console.WriteLine ile birden çok sayı gösterecekseniz bu durumda string birleştirme işleminden yararlanabilirsiniz. String birleştirme işlemi string veri türü yazısında daha önce örneklerle görmüştük.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4;
        Console.WriteLine(A + " ve " + B + " sayıları toplamı "
            + (A + B) + " eder.");
    }
}
```

Yukarıdaki örnekte gördüğümüz gibi karmaşık bir yazı şekli oldu. Bunun daha güzel bir şekli vardır. Bu da yer belirtme yöntemidir.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4;
        Console.WriteLine("{0} ve {1} sayıları toplamı {2} eder",
            A, B, A + B );
    }
}
```

Yukarıdaki örnekte WriteLine yöntemi dört tane parametre almıştır. İlk parametre ekranda gösterilecek olan string ifadedir. Bu string ifade içerisinde yer tutucular var ve dizilerde olduğu gibi 0 ile başlamıştır, küme parantezleri arasında belirtilmiştir. Bu yer tutucular gösterilecek olan string ifadeyi takip eden virgülle ayrılmış diğer üç tane parametre için yer belirtmektedir. {0} ile belirtilen yere A değişkeninin değeri, {1} ile belirtilen yere B değişkeninin değeri, {2} ile belirtilen yere de A+B işleminin sonucu gelecektir. İlk parametreden sonra belirtilecek parametreler sayısal ifadeler olabileceği gibi string ifadeler, string literaller de olabilir.

Eğer yazının içerisinde küme parantezi geçecekse bunu ardı ardına iki tane kullanarak yapmalıyız.

Bu yer tutucuları yazı içerisinde sıra ile yazmak zorunda değildir. Nerede bulunmaları gerekiyorsa oraya koyabiliriz. Aşağıdaki örnekte yer tutucular {0} {1} {2} sırasıyla yazılmamıştır.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4;
        Console.WriteLine("{2} değeri {0} ve {1} sayılarının toplamıdır.",
            A, B, A + B );
    }
}
```

İlk parametre olan string ifadeden sonra yazacağımız parametre sayısı yer tutuculardan fazla olabilir. Bir parametreyi yazmışsak ve onun için yer tutucu belirtmemişsek problem olmaz.

Aşağıdaki örnekte yazılan C parametresi hiç kullanılmamıştır ama programın çalışması için bir engel teşkil etmez.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4, C=20;
        Console.WriteLine("{2} deęeri {0} ve {1} sayılarının toplamıdır.",
            A, B, A + B, C );
    }
}
```

Aşğıdaki örnekte olduęu gibi 4 tane parametre düşünelim. 0, 1 ve 3 numaralı parametreleri kullanacağız ve 2 numaralı parametre hiç kullanılmayacak. Böyle bir kullanım da mümkündür. Kullanılmayacak olan parametrenin en son yazılma zorunluluęu da yoktur.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4, C=20;
        Console.WriteLine("{3} deęeri {0} ve {1} sayılarının toplamıdır.",
            A, B, C, A + B);
    }
}
```

Bir yer tutucuyu tekrar tekrar kullanmak da mümkündür.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4, C=20;
        Console.WriteLine("{3} deęeri {0} ve {1} sayılarının toplamıdır.
            Sonuç {3} deęeridir.",
            A, B, C, A + B);
    }
}
```

Ama olmayan bir parametre için yer tutucu yazmak doğaldır ki yanlıştır ve bir istisna ortaya çıkacaktır.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4, C=20;
        Console.WriteLine("{0} ve {1} sayıları toplamı {3} eder",
            A, B, A + B);
    }
}
```

```
C:\WINDOWS\system32\cmd.exe
Unhandled Exception: System.FormatException: Index (zero based) must be greater than or equal to zero and less than the size of the argument list.
   at System.Text.StringBuilder.AppendFormat(IFormatProvider provider, String format, Object[] args)
   at System.String.Format(IFormatProvider provider, String format, Object[] args)
   at System.IO.TextWriter.WriteLine(String format, Object arg0, Object arg1, Object arg2)
   at System.IO.TextWriter.SyncTextWriter.WriteLine(String format, Object arg0, Object arg1, Object arg2)
   at System.Console.WriteLine(String format, Object arg0, Object arg1, Object arg2)
   at NumerikFormatlama.Main() in C:\Documents and Settings\KOMUR ALP\Belgelerim\Visual Studio 2005\Projects\fancyformatting örneği\fancyformatting örneği\Program.cs:line 7
Devam etmek için bir tuşa basın . . .
```

Yukarıda string birleştirme işlemi ile yazdığımız örneği bir defa daha yazalım ama açık ToString yöntem çağruları ile biçimlendirme de yapalım.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4;
        Console.WriteLine(A.ToString("E4") + " ve " +
            B.ToString("E4") + " sayıların toplamı "
            + (A + B).ToString("E4") + " eder.");
    }
}
```

Karmaşık oldu, yer tutucular kullanarak yazalım.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4;
        Console.WriteLine("{0} ve {1} sayıların toplamı {2} eder",
            A.ToString("E4"), B.ToString("E4"), (A + B).ToString("E4"));
    }
}
```

Ama gene karmaşık oldu. Daha güzel bir yöntem var. Sayısal biçimlendirme parametrelerini yer tutucularla birlikte yazabiliriz. Açık ToString yöntem çağrularına da gerek kalmaz.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        int A = 3, B = 4;
        Console.WriteLine("{0:E4} ve {1:E4} sayıların toplamı {2:E4} eder",
            A, B, A + B);
    }
}
```

```
}
```

Yer tutucu numarası bir iki nokta işareti ve ardından biçimlendirme parametresi geliyor. Bu parametreler normalde ToString yöntemine geçiliyor, ama bu şekilde kullanım çok daha pratik olduğu için daha çok tercih ediliyor.

Çoğu durumda bir ifadenin karakter olarak ne kadar yer kaplayacağını bilmek imkânsızdır, meselâ tamsayılar 10 karaktere kadar, eğer negatif değer alabiliyorlarsa 11 karaktere kadar, üçer basamak ayrılıyorsa 4 karaktere kadar yer kaplayabilirler. Aşağıdaki örnekte sayılar iç içe geçmektedir.

```
Console.WriteLine("{0}{1}{2}{3}", A, B, C, D);
```

Bunu kolaylıkla aralarına boşluklar bırakarak ayırabiliriz.

```
Console.WriteLine("{0} {1} {2} {3}", A, B, C, D);
```

Alt alta birden çok WriteLine değeri yazdığımızı düşünürseniz belirli yerler kaplayarak sütunlar halinde alt alta yazılmasını isteyebiliriz. Alan genişliği denilen sayılar yazarak ne kadar yer kaplayacaklarını belirtebiliriz, bunu da yer tutucu numaraların yanına virgöl koyup akabinde alan genişliği değerini giriyoruz.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        decimal mA = 123.45m;
        string strB = "Kitap";
        int iC = 12;
        decimal mD = 456678m;
        Console.WriteLine("{0,10}{1,15}{2,10}{3,20}", mA, strB, iC, mD);
    }
}
```

mA sayısı sağa dayalı olarak 10 karakter, strB değeri yine sağa dayalı olarak 15 karakter, iC değeri yine sağa dayalı olarak 10 karakter, mD değeri yine sağa dayalı olarak 20 karakter yer kullanılarak gösterildi.

Çıktısı da aşağıdaki şekildedir.

Konsolda bir satırda gösterebileceğimiz karakter sınırı da yine IBM punch kartlarındaki sınır olan 80 karakterdir.

Biçimleme parametreleri ile alan genişliği parametrelerini birlikte kullanabilirsiniz.

```
using System;
class NumerikFormatlama
{
    static void Main()
    {
        decimal mA = 123.45m;
        string strB = "Kitap";
        int iC = 12;
        decimal mD = 456678m;
        Console.WriteLine("{0,-15}{1,10:C2}{2,10:X5}{3,20:E3}", strB, mA, iC, mD);
    }
}
```

}

Yukarıdaki örnekte {0} yer tutucusunun gösterdiği değer dikkat ederseniz alan genişliği olarak -15 değerini almıştır. Bu da 15 karakterlik alanda soldan hizalama yapılacağı anlamına gelmektedir. Hem biçimlendirme hem de alan belirleme parametreleri birlikte kullanıldı.