Bölüm 26

Çıktıyı Biçemli Yazdırma

Neden Biçemleme
Sayıların Biçemlenmesi
`Write()` ya da `WriteLine()` metotları ile çıktı elde etme
Sağa/sola Yanaşık Yazdırma
{} Yer Tutucu ile Biçemleme
Standart Sayısal Biçem Belirtgenleri
Simgesel biçimler (picture formats)
Üstel Notasyon
`ToString()` metodunu kullanmak
`NumberFormatInfo`

Neden Biçemleme

Bilgisayar çıktısının kolay okunur ve kolay anlaşılır biçeme (format) konulması uygulamada önem taşır. Özellikle, sayılar ve tarihler farklı ülke, farklı alfabe ve farklı kültürlerde farklı biçimlerde yazılır. Örneğin, biz kesirli sayıların kesir kısmını (ondalık kısım) `kesir ayıracı` dediğimiz virgül (,) ile ayırırız. Tam kısmı çok haneli olan sayıları kolay okumak için, sayının tam kısmının hanelerini sağdan sola doğru üçerli gruplara ayırırız. Bu işe sayıyı binliklerine ayırmak diyoruz. Örneğin,

```
123.456,789
```

sayısı, bizim için tamsayı kısmı yüz yirmi üç bin dörtyüz elli altı ve kesirli sayı (ondalık) kısmı binde yedi yüz seksen dokuz olan bir sayıdır. Ama aynı anlama gelmesi için bu sayı İngiltere'de veya ABD'de

```
123,456.789
```

biçiminde yazılmalıdır.

C# dili java dilinde olduğu gibi, karakterleri 16 bitlik `Unicode` sistemiyle yazar. Dolayısıyla, dünyadaki bütün dillere, alfabelere ve kültürlere hizmet edebilme yeteneğine sahip gelişkin bir dildir. Bu demektir ki, C# dili metinleri ve sayıları istediğimiz alfabede ve istediğimiz biçemde (format) yazabilir.

Bu kitap, konuları sistematik anlatmak yerine pedagojik anlatma yöntemini seçmiştir. O nedenle, çıktıların nasıl biçemlendiğinin sistematiğine çok girmeden, pratik uygulamalarla konuyu açıklayacağız.

Sayıların Biçemlenmesi

C# dilinde `tamsayı` veri tiplerini 8, `kesirli sayı` tiplerini de 3 ayrı gruba ayırmıştık. `.NET` onların herbirisini `System` aduzayında (namespace) içinde ayrı birer sınıf olarak tanımlar. Dolayısıyla her bir sınıfın kendisine özgü metotları vardır.

Çıktıyı biçimlendirmek için üç yöntem kullanırız.

1. `System.Console` sınıfı içindeki `Write()` ya da `WriteLine()` metotlarını kullanmak.
2. Her sınıfta var olan `ToString()` metodunu kullanmak.
3. `String.Format()` metodunu kullanmak.

Özellikle, hemen her sınıfta ortak ad taşıyan `ToString()` metodunun sayısal veri tiplerinin (sınıflarının) hepsinde olduğunu unutmayalım. Bu metodu kullanarak, sayıların yerel kültürlere uyan çıktılarını elde edebiliriz. Bunu yaparken sayısal verileri string'e dönüştürerek biçimlendirmiş oluyoruz. Başka bir deyişle, sayıyı temsil eden stringi biçimlendiriyoruz.

Çıktıyı biçimlendirmek için kullandığımız `String.Format()` metodu ile hemen hemen her kültüre uyan çıktı biçemleri elde edebiliriz.

Şimdi bu üç yöntemi örnekler üzerinde açıklamaya başlayalım.

`Write()` ya da `WriteLine()` metotları ile çıktı elde etme

En basit sayısal çıktıyı `System.Console` sınıfı içindeki `Write()` ya da `WriteLine()` metotları ile elde ederiz. Parametre yazarken tamsayıları olduğu gibi yazarız. Kesirli sayılarda kesir ayracı olarak (.) simgesini kullanarak yazarız. Konsola giden çıktı, o an bilgisayarı çalıştıran Windows İşletim Sisteminde etkin olan dil ve kültüre uyacak biçimde kendiliğinden biçimlenir. Tamsayılar olduğu gibi çıkar. Kesirli sayılar için girdi ile çıktı arasında yöresellik farkı oluşur. Türkçe Windows işletim sisteminde 1234.5678 biçiminde yazılan parametre, Türkçe'de kullandığımız 1234,5678 biçimiyle çıkar. Burada dikkat edilecek tek nokta, kesirli sayıları parametre olarak kullanılırken kesir ayracının (,) değil (.) olması gerektiğidir. Parametre girişi standarttır, ama konsol çıktısı dil ve kültüre göre değişir. Aşağıdaki program bir tamsayı ve bir kesirlisayı giriş ve çıkışını göstermektedir.

Biçem01.cs

```
using System;

namespace SayısalBiçemler
{
    class Sayılar01
    {
        static void Main(string[] args)
        {
            Console.WriteLine(12345678);
            Console.WriteLine(1234.5678);
        }
    }
}
```

Çıktı

```
12345678
1234,5678
```

Sayıları doğrudan parametre olarak kullanmak yerine, onları tutan değişken adlarını parametre olarak kullanabiliriz. Bu durumda da yukarıda söylediklerimiz geçerlidir.

Biçem02.cs

```
using System;

namespace SayısalBiçemler
{
    class Sayılar01
    {
        static void Main(string[] args)
        {
            int  tamSayı = 12345678;
            double kesirliSayı = 1234.5678;
            Console.WriteLine(tamSayı);
            Console.WriteLine(kesirliSayı);
        }
    }
}
```

Çıktı

```
12345678
1234,5678
```

Sağa/sola Yanaşık Yazdırma

string'lerde olduğu gibi tamsayıları belli bir alana sağa ya da sola yanaşık yazdırabiliriz. Bunun için `{}` `yer tutucusu`'nu kullanırız. `{n,xx}` yer tutucusundaki `n` sayısı 0,1,2,... sayılarından birisidir. Kaçıncı değişkene yer tutulacağını belirtir. `xx` pozitif tamsayısı değişken değerinin kaç haneye sağa yanaşık olarak yazılacağını belirtir. `-xx` sayısı değişken değerinin kaç haneye sola yanaşık olarak yazılacağını belirtir.

Biçem04.cs

```
using System;

namespace SayılarıBiçemleme
{
    class Sayılar
    {
        static void Main(string[] args)
        {
            Console.WriteLine("|{0,-20}|", 12345678);
            Console.WriteLine("|{0, 20}|", 12345678);
        }
    }
}
```

Çıktı

```
|12345678        |
|        12345678|
```

{} Yer Tutucu ile Biçemleme

`{ }` değişkene `yer tutucu`'dur. Bunun içine *basamak tutucuları* dediğimiz `'#'` ve `'0'` karakterlerini, sayıyı görmek istediğimiz biçemde yerleştiririz. `'#'` basmak tutucusu ancak sayıya gerektiği kadar basamak ayarlar. `'0'` basamak tutucusu ise, sayıda olmayan basamaklar yerine `'0'` koyar. Bunların işlevlerini 9807605,4361 sayısını farklı biçemlerde yazdırak görelim.

Biçem05.cs

```
using System;

public class ExponentialNotation
{
    public static void Main()
    {
        double dSayı = 9807605.4361;
        Console.WriteLine(dSayı);
        Console.WriteLine("{0}", dSayı);

        Console.WriteLine();
        Console.WriteLine("{0:0.00}", dSayı);
        Console.WriteLine("{0:#.##}", dSayı);

        Console.WriteLine();
        Console.WriteLine("{0:000,000,000.0000000}", dSayı);
        Console.WriteLine("{0:###,###,###.#######}", dSayı);

        Console.WriteLine();
        Console.WriteLine("{0:000000000.0000000}", dSayı);
        Console.WriteLine("{0:#########.#######}", dSayı);

        Console.WriteLine();
        Console.WriteLine("{0:##.##}", dSayı);
        Console.WriteLine("{0:00.00}", dSayı);

        Console.WriteLine();
        Console.WriteLine("{0:00.###}", dSayı);
        Console.WriteLine("{0:##.0}", dSayı);
    }
}
```

Çıktı

```
9807605,4361
9807605,4361

9807605,44
9807605,44

009.807.605,4361000
9.807.605,4361

009807605,4361000
9807605,4361
9807605,44
9807605,44
```

```
9807605,446
9807605,4
```

Bu çıktıyı satır satır incelersek, `'#'` ve `'0'` basamak tutucularının yaptıklarını hemen görebileceğiz.

1 ve 2 nci satırı veren deyimler, esasta aynıdırlar.

3 ve 4 üncü satırlarda, sayının tam kısmının basamak sayısı istenen biçemdekinden fazladır. Dolayısıyla, C# , isteğe uymaz, sayının tam kısmının basamaklarını eksiksiz yazar. Kesirli kısım iki haneli istenmiştir. Sayının kesir kısmı iki haneye yuvarlanır.

5 ve 6 ıncı satırlar için istenen biçemde, sayının tam ve kesirli basamakları sayısından fazla basamak tutucusu konmuştur. Bu durumda `'#' basamak tutucusu` gerektiği kadar haneyi kullandırır. `'0' basamak tutucusu` ise, boş kalan basamakları '0' karakteri ile doldurur.

7 ve 8 inci satırlar için istenen basamak sayıları 5-6 ıncı satırlardaki gibidir. Ancak, sayının tam kısmı binliklerine gruplanmamıştır. Dolayısıyla, çıktı, binlik gruplara ayrılmamış olarak gelmektedir.

9-10-11-12 inci satırlar için istenen biçemdeki tamsayı basamakları yetmediğinden, C# , o isteğe uymaz, sayının tam kısmını eksiksiz yazar. Kesirli kısım için istenen hane sayısı sayınınkinden az olduğundan, sayının kesir kısmı yuvarlanarak yazılır.

Standart Sayısal Biçem Belirtgenleri

Sayısal verileri biçimlendirmek için kullanılan metinler (strig) dir. Örneğin, finansal uygulamalarda para ifade eden sayıların önüne veya arkasına para biriminin yazılması ve çıktının kaç haneye yazılacağının belirtilmesi standart bir sayısal biçem belirtgenidir. `c12` belirtgeni sayının önüne para biriminin konulacağını ve çıktının 12 haneye sağa yanaşık yazılacağını belirtir. Belirgende ilk harf olarak `C,c, D,d, E,e, F,f, G,g, N,n, P,p, R,r, X,x` harflerinden birisi konulur. Bu harflerin işlevleri aşağıdaki tabloda belirtilmiştir.

Sayısal Biçem Belirtgenleri (Numeric Format Specifiers)

Belirteç	Biçem	Çıktı
C[n] ,c[n]	$XX,XX.XX ($XX,XXX.XX)	Para (Currency)
D[n]	[-]XXXXXXX	Sayısal (decimal)
E[n] , e[n]	[-]X.XXXXXXE+xxx [-]X.XXXXXXe+xxx [-]X.XXXXXXE-xxx [-]X.XXXXXXe-xxx	Üstel (Exponent)
F[n]	[-]XXXXXXX.XX	Kesir ayracı (Fixed point)
G[n]	General or scientific	Genel
N[n]	[-]XX,XXX.XX	Sayı
P[p]		Yüzde
R[r]		Dönüşlü (round-trip)
X[n] , x[n]	Hex representation	Hexadecimal

`R, r` : Bir sayının stringe dönüşmesi halinde tekrar aynı sayıya dönüşebileceğini garanti eder.

Aşağıdaki programlar tamsayılarda ve kesirli sayılarda sayısal biçim belirteçlerinin etkilerini göstermektedir. Satırların karşısında yazılan çıktıları dikkatle inceleyiniz.

Biçem06.cs

```
using System;
class TestDefaultBiçemler
{
    static void Main()
    {
        int i = 123456789;
        Console.WriteLine("{0:C}", i); // 123.456.789,00TL
        Console.WriteLine("{0:D}", i); // 123456789
        Console.WriteLine("{0:E}", i); // 1,2345678E+008
        Console.WriteLine("{0:F}", i); // 123456789,00
        Console.WriteLine("{0:G}", i); // 123456789
        Console.WriteLine("{0:N}", i); // 123.456.789,00
        Console.WriteLine("{0:X}", i); // 75BCD15
        Console.WriteLine("{0:x}", i); // 75bcd15,
    }
}
```

Biçem07.cs

```
using System;
class TestIntegerFormats
{
    static void Main()
    {
        int i = 123;
        Console.WriteLine("{0:C6}", i); // 123,000000TL
        Console.WriteLine("{0:D6}", i); // 000123
        Console.WriteLine("{0:E6}", i); // 1,230000E+002
        Console.WriteLine("{0:G6}", i); // 123
        Console.WriteLine("{0:N6}", i); // 123,000000
        Console.WriteLine("{0:X6}", i); // 00007B
        i = -123;
        Console.WriteLine("{0:C6}", i); // -123,000000TL
        Console.WriteLine("{0:D6}", i); // -000123
        Console.WriteLine("{0:E6}", i); // -1,230000E+002
        Console.WriteLine("{0:G6}", i); // -123
        Console.WriteLine("{0:N6}", i); // -123,000000
        Console.WriteLine("{0:X6}", i); // FFFFF85
        i = 0;
        Console.WriteLine("{0:C6}", i); // 0,000000TL
        Console.WriteLine("{0:D6}", i); // 000000
        Console.WriteLine("{0:E6}", i); // 0,000000E+000
        Console.WriteLine("{0:G6}", i); // 0
        Console.WriteLine("{0:N6}", i); // 0,000000
        Console.WriteLine("{0:X6}", i); // 000000
    }
}
```

Biçem08.cs

```
using System;
class TestDoubleFormats
{
    static void Main()
```

```
    {
        double d = 1.23;
        Console.WriteLine("{0:C6}", d); // 1,230000Tl
        Console.WriteLine("{0:E6}", d); // 1,230000E+000
        Console.WriteLine("{0:G6}", d); // 1,23
        Console.WriteLine("{0:N6}", d); // 1.230000
        d = -1.23;
        Console.WriteLine("{0:C6}", d); // -1,230000TL
        Console.WriteLine("{0:E6}", d); // -1,230000E+000
        Console.WriteLine("{0:G6}", d); // -1,23
        Console.WriteLine("{0:N6}", d); // -1,230000
        d = 0;
        Console.WriteLine("{0:C6}", d); // 0,000000TL
        Console.WriteLine("{0:E6}", d); // 0,000000E+000
        Console.WriteLine("{0:G6}", d); // 0
        Console.WriteLine("{0:N6}", d); // 0,000000
    }
}
```

Simgesel biçemler (picture formats)

Belirteç	Sonuç (string)	C# ifadesi
0	Sıfır basamak tutucu	Zero placeholder
#	Rakam basamak tutucu	Digit placeholder
.	Kesir ayracı	Decimal point
,	grup ayracı	Group separator or multiplier
%	Yüzde simgesi	Percent notation
E+0, E-0 e+0, e-0	Üstel notasyon	Exponent notation
\	Literal karakter belirtme	Literal character quote
'xx'"xx"	Literal string belirtme	Literal string quote
;	Bölüm ayracı	Section separator

Biçem09.cs

```
using System;
class TestIntegerCustomFormats
{
    static void Main()
    {
        int i = 123;
        Console.WriteLine("{0:#0}", i); // 123
        Console.WriteLine("{0:#0;(#0)}", i); // 123
        Console.WriteLine("{0:#0;(#0);<zero>}", i); // 123
        Console.WriteLine("{0:#%}", i); // 12300%
        i = -123;
        Console.WriteLine("{0:#0}", i); // -123
        Console.WriteLine("{0:#0;(#0)}", i); // (123)
        Console.WriteLine("{0:#0;(#0);<zero>}", i); // (123)
        Console.WriteLine("{0:#%}", i); // -12300%
        i = 0;
        Console.WriteLine("{0:#0}", i); // 0
        Console.WriteLine("{0:#0;(#0)}", i); // 0
```

```
        Console.WriteLine("{0:#0;(#0);<zero>}", i); // <zero>
        Console.WriteLine("{0:#%}", i); // %
    }
}
```

Alıştırmalar

Biçem10.cs

```
// Biçemleme örnekleri

using System;

public class PictureFormatDemo
{
    public static void Main()
    {
        double dSayı = 98765.10234;

        Console.WriteLine("Default biçem: " + dSayı);

        // 2 ondalık kesirli
        Console.WriteLine("İki kesirli hane: " +
                          "{0:#.##}", dSayı);

        // Binliklere ayır.
        Console.WriteLine("Binliklere ayrılmış: {0:#,###.##}", dSayı);

        // Bilimsel notasyon
        Console.WriteLine("Bilimsel notasyon: " +
                          "{0:#.###e+00}", dSayı);

        // Scale the value by 1000.
        Console.WriteLine("Value in 1,000s: " +
                          "{0:#0,}", dSayı);

        /* Pozitif, negatif ve sıfır
           sayılarının farklı gösterimleri */

        Console.WriteLine("Pozitif, negatif ve sıfır  sayılarının farklı
gösterimleri");
        Console.WriteLine("Pozitif sayı : {0:#.#;(#.##);0.00}", dSayı);
        dSayı = -dSayı;
        Console.WriteLine("Negatif Sayı: {0:#.##;(#.##);0.00}", dSayı);
        dSayı = 0.0;
        Console.WriteLine("Sıfır Sayısı: {0:#.##;(#.##);0.00}", dSayı);

        // Yüzde gösterimi.
        dSayı = 0.17;
        Console.WriteLine("Yüzde gösterimi: {0:#%}", dSayı);
    }
}
```

Çıktı

```
Default biçem: 98765,10234
İki kesirli hane: 98765,1
Binliklere ayrılmış: 98.765,1
```

```
Bilimsel notasyon: 9,877e+04
Value in 1,000s: 99
Pozitif, negatif ve sıfır  sayılarının farklı gösterimleri
Pozitif sayı : 98765,1
Negatif Sayı: (98765,1)
Sıfır Sayısı: 0,00
Yüzde gösterimi: 17%
```

Yer tutucuda sayının hanelerinden daha az ya da daha çok # hanesi yazılması, tamsayıyı değiştirmez. Çıktıda her zaman tamsayıya yetecek kadar hane ayrılır.

Biçem11.cs

```
// Biçemleme örnekleri

using System;

public class YerTutucuBiçemlemeleri
{
public class Resimleme
{
    public static void Main()
    {
        Console.WriteLine("{0:#####}", 123);
        Console.WriteLine("{0:#####}", 1234);
        Console.WriteLine("{0:###}", 12345);
    }
}
}
```

Çıktı

```
123
1234
12345
```

Çıktıda özellikle kesir haneleri istenmediğinde, kesirli sayı en yakın tamsayıya yuvarlanır.

Biçem12.cs

```
// Biçemleme örnekleri

using System;

public class YerTutucuBiçemlemeleri
{
public class Resimleme
{
    public static void Main()
    {
        Console.WriteLine("{0:#####}", 123.45678);
        Console.WriteLine("{0:#####}", 1234.56789);
        Console.WriteLine("{0:###}", 12345.908765);
    }
}
}
```

Çıktı

```
123
1235
12346
```

Çıkış biçeminde kaç tane kesirli sayı hanesi istediğimizi belirtebiliriz. İstenen hane sayısı sayının kesirli hanelerinden çok ise, boş kalan haneler sağa doğru 0 ile dolar. İstenen hane sayısı sayının kesirli hanelerinden az ise, sayının kesir kısmı yuvarlanarak alınır. Sayının tamsayı kısımları, istenen hane sayısından asla etkilenmez. Sayının tamsayı haneleri çıktıda aynen görünür.

Biçem13.cs

```
using System;

public class KesirAyracı
{
    public static void Main()
    {
        Console.WriteLine("{0:#####.000}", 98765.6);
        Console.WriteLine("{0:##.000}", 8765.654321);
    }
}
```

Çıktı

```
98765,600
8765,654
```

Üstel Notasyon

Her sayıyı üstel notasyonla yazabiliriz. Örneğin, `12345 = 1,2345x10^5` dir. Bu sayının konsola giden çıktısı `1,2345E+5` biçiminde gösterilir. `E+5` yerine `E5, E05, E005, E+05, E+005` gibi notasyonlar da kullanılır. Burada `E+5` ifadesi `(x10^5)` yerine geçer. E nin solunda kalan sayının 10 üssü 5 ile çarpılacağı anlamına gelir. E harfinden sonra gelen sayı `" 10 nun kuvveti"` veya `"10 nun üssü"` adını alır. Bu üs `bir, iki, üç, ...` haneli yazılabilir. Çünkü `5, 05, 005,...` sayıları aynıdır.

Üstel notasyonda, sayının tam kısmının kaç basamaklı olacağı, kesirli kısma kaç basamak konacağı, üstel sayının nasıl gösterileceği `{} yer tutucusu` içinde belirlenebilir. İstenen kesir basamaklarının sayısı, sayının kesir basamaklarından az değilse, kesir basamakları aynen gösterilir. İstenen kesir basamakları sayısı sayınınkinden az ise, sığmayan basamaktan sonrası yuvarlanır.

Biçem14.cs

```
"{0:#.000E-00}"
using System;

public class ÜstelNotasyon
{
    public static void Main()
    {
        double pi = 3.14159265358979323846264338327950288419716939937510;

        Console.WriteLine("{0:#.000E-00}", pi);
        Console.WriteLine("{0:0.000E-00}", pi);
        Console.WriteLine("{0:0.000E+00}", pi);
```

tSayı null

```
        Console.WriteLine("{0:#.0000000E+000}", pi);
        Console.WriteLine("{0:0.0000000E+000}", pi);
        Console.WriteLine("{0:#.0000000E-000}", pi);

        Console.WriteLine("{0:##.000E-00}", pi);
        Console.WriteLine("{0:#######.0000000E+000}", pi);
        Console.WriteLine("{0:#######.0000000E-000}", pi);
        Console.WriteLine("{0:#,###,###.0000000E+000}", pi);
    }
}
```

Çıktı

```
3,142E00
3,142E00
3,142E+00
3,1415927E+000
3,1415927E+000
3,1415927E000
31,416E-01
3141592,6535898E-006
3141592,6535898E-006
3.141.592,6535898E-006
```

`ToString()` metodunu kullanmak

C# dilinde 8 tane tamsayı sınıfı ile 3 tane kesirli sayı sınıfının var olduğunu söylemiştik. Bu sınıfların hepsinde `ToString()` metodu vardır. Bu metodun görevi, bir nesne olarak tutulan sayıyı istenilen biçeme (string) dönüştürmektir. Stringe dönüşen sayıyı, `Write()` veya `WriteLine()` metotları konsola gönderir.

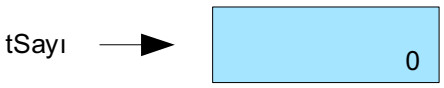
Aşağıdaki örnek `ToString()` metodunun kullanılışını göstermektedir. `ToString()` metodu bir tane değildir. Sayı sınıflarının hepsinde bir `ToString()` metodu vardır. Dolayısıyla, yazdıracağımız sayı hangi sınıfa aitse o sınıfın bir nesnesini yaratmamız gerekir. Başka bir deyişle, sayıyı o sınıfa ait bir nesne olarak yaratmak gerekir.

```
int tSayı ;
tSayı = new int();
tSayı = 12345678;
Console.WriteLine(tSayı.ToString());
```

deyimlerini bire birer inceleyelim.

```
int tSayı ;
```

deyimi, `int` sınıfına ait `tSayı` adlı bir nesne işaretçisinin bildirimidir. `tSayı`, int sınıfına ait bir nesnenin bellekteki adresini gösterecektir. Bu nedenle, ona işaretçi, referans, pointer sıfatlarını veririz. C/C++ dillerinde "pointer" sıfatı kullanılır. C# dilinde *"pointer"* sözcüğü yerine *"referans"* sözcüğü kullanılır. Biz *"referans"*, *"işaretçi"* ve *"pointer"* sıfatlarını eşanlamlı olarak kullanacağız. Henüz bu aşamada işaret edilecek nesne yaratılmamıştır. Dolayısıyla, `tsayı` işaretçisinin gösterdiği yer `'null'`dır; yani bellekte bir adres göstermez. Bir nesneyi işaret edebilmesi için, öncelikle o nesnenin yaratılmış olması gerekir. Bundan sonraki deyim onu yapmaktadır.




```
tSayı = new int();
```

deyimi int tipinden bir nesne yaratır; yani bellekte `int` tipi verinin sığacağı bir yer açar. Nesne Yönelimli programlamada bu eyleme nesne yaratma (constructor, instantiate) denir. Bellekte açılan o yere nesnenin adresi diyoruz. `tSayı` işaretçisi yaratılan nesnenin bellekteki adresini göstermeye başlar. Henüz bu aşamada o adrese bir değer girilmemiştir. C# sayısal nesneler yaratılır yaratılmaz onlara kendiliğinden 0 değerini (default value) atar. Bu değer nesneye başka değer atanınca değişir.

tSayı = 12345678;

deyimi, yaratılan nesneye 12345678 değerini atar. Bu eylem, 12345678 sayısının açılan adrese girmesi demektir. O sayı artık bir `int` nesnesidir. `int` sınıfına ait bütün özelikleri taşır, ona `int` sınıfına ait metotlar uygulanabilir.

Dördüncü satırdaki

```
Console.WriteLine(tSayı.ToString());
```

deyimi ard arda iki iş yapmaktadır. `tSayı.ToString()` metodu `tSayı` işaretçisinin işaret ettiği nesneyi stringe dönüştürür. `WriteLine()` metodu onu konsola yazar.

Benzer işleri kesirli sayılar için de yaparız. Aşağıdaki kodlar o işi yapmaktadır.

```
double kSayı = new double();
kSayı = 1234.5678;
Console.WriteLine(kSayı.ToString());
```

Yukarıda iki satırda yapılan iş burada ilk satıra tek deyim olarak yazılmıştır. Bu deyim double sınıfına ait bir nesnenin bellekteki adresini gösterecek kSayı işaretçisinin bildirimini yapmakta ve onun göstereceği nesneye bellekte bir yer ayırmaktadır. İkinci ve üçüncü satırın işlevleri yukarıda açıkladıklarımıza benzer.

Bütün bu söylediklerimizi aşağıdaki programa koyalım.

Biçem15.cs

```
using System;

namespace SayılarıBiçemleme
{
    class Sayılar
    {
        static void Main(string[] args)
        {
            int tSayı ;
            tSayı = new int();
            tSayı = 12345678;
            Console.WriteLine(tSayı.ToString());

            double kSayı = new double();
            kSayı = 1234.5678;
            Console.WriteLine(kSayı.ToString());
        }
    }
}
```

Çıktı

```
12345678
1234,5678
```

1.

Biçem16.cs

```
using System;

namespace SayılarıBiçemleme
{
    public class ToStringBiçemi
    {
        public static void Main()
        {
            int n = 123;
            double kks = 0.35;
            double bks = 12345.67809;

            string str = bks.ToString("F2");
            Console.WriteLine(str);

            str = bks.ToString("N5");
            Console.WriteLine(str);

            str = bks.ToString("e");
            Console.WriteLine(str);

            str = bks.ToString("r");
            Console.WriteLine(str);

            str = kks.ToString("p");
            Console.WriteLine(str);

            str = n.ToString("X");
            Console.WriteLine(str);

            str = n.ToString("D12");
            Console.WriteLine(str);

            str = 189.99.ToString("C");
            Console.WriteLine(str);
        }
    }
}
```

Çıktı

```
12345,68
12.345,67809
1,234568e+004
12345,67809
%35,00
7B
000000000123
189,99 TL
```

NumberFormatInfo

Bu sınıf, kültürlerden bağımsız biçemler yaratır. Biçemleme seçeneklerini değiştirebilir. Örneğin, pozitif sayıların, yüzde simgesinin, binliklere ayırma ayracının, para biriminin nasıl gösterileceği gibi şeyleri onunla yapabiliriz.

Bu söylediklerimizin nasıl yapıldığını örnekler üzerinde açıklayacağız. Aşağıdaki ilk örnekte 1234567890 sayısının çıktısı *Türkçe, Fransızca* ve *Almanca* biçemleriyle verilmektedir.

Biçem17.cs

```
using System;
using System.Globalization;
using System.Threading;

namespace Yöresellik
{
    class Yerellik01
    {
        static void Main(string[] args)
        {
            int birSayı = 1234567890;

            // mevsut thread'deki yerel (Türkçe)biçim
            Console.WriteLine(birSayı.ToString("N"));

            // IFormatProvider kullanımı ile Fransızca
            Console.WriteLine(birSayı.ToString("N",
                              new CultureInfo("fr-FR")));

            // kültür thread'inin değiştirilmesi (Almanca)
            Thread.CurrentThread.CurrentCulture =

                              new CultureInfo("de-DE");
            Console.WriteLine(birSayı.ToString("N"));
        }
    }
}
```

Çıktı

```
1.234.567.890,00
1 234 567 890,00
1.234.567.890,00
```

İlk satırı yazan

```
Console.WriteLine(birSayı.ToString("N"));
```

deyimidir.

```
birSayı.ToString("N")
```

metodu `birSayı` nesnesini `N biçemine` dönüştürerek standart çıkışa (konsol) gönderiyor. N biçemi, o anda etkin olan kültür thread'ine göre sayının yazılış biçimidir. Bilgisayarımız Türkçe Windows işletim sistemiyle çalıştığı için, doğal (default) kültür thread'i Türkçe'dir. Dolayısıyla, programdaki ilk `ToString("N")` metodu `birSayı` nesnesini (ki o 1234567890 sayısıdır) Türkçe biçemli string'e dönüştürecektir. `Console.WriteLine()` metodu ise, o dönüşmüş string'i konsola (standart çıkış) yazar.

İkinci satırı yazan

```
Console.WriteLine(birSayı.ToString("N", new CultureInfo("fr-FR")));
```

deyimidir. Burda ToString() metodunun ikinci parametresi

```
new CultureInfo("fr-FR")
```

deyimi ile yaratılan CultureInfo sınıfına ait bir nesnedir. Bu nesne "fr-FR" parametresi ile Fransız kültür thread'ini almış olur. Dolayısıyla,

```
ToString("N", new CultureInfo("fr-FR"))
```

metodu birSayı nesnesini Fransızca'daki sayı yazma biçemiyle string'e dönüştürür.

Üçüncü satır da benzer olarak Almanca sayı yazma biçemine dönüşmektedir.

Biçem18.cs

```
// String.Format() kullanımına örnek
using System;

namespace SayılarıBiçemleme
{
    public class FormatDemo2
    {
        public static void Main()
        {
            int i;
            int toplam = 0;
            int çarpım = 1;
            string str;

            /* Program koşarken 1-10 arasındaki sayıların
               toplmı ve çarpımını listeler */
            for (i = 1; i <= 10; i++)
            {
                toplam += i;
                çarpım *= i;
                str = String.Format("Toplam:{0,3:D}  Çarpım:{1,12:D}",
                                    toplam, çarpım);
                Console.WriteLine(str);
            }
        }
    }
}
```

Çıktı

```
Toplam:  1  Çarpım:          1
Toplam:  3  Çarpım:          2
Toplam:  6  Çarpım:          6
Toplam: 10  Çarpım:         24
Toplam: 15  Çarpım:        120
Toplam: 21  Çarpım:        720
Toplam: 28  Çarpım:       5040
Toplam: 36  Çarpım:      40320
Toplam: 45  Çarpım:     362880
Toplam: 55  Çarpım:    3628800
```

Sorular

1. Sayıları 5 haneye ön boşluklara 0 lar koyarak nasıl yazdırılır?

```
int _num1 = 123;
int _num2 = 45;
int _num3 = 123456;
String.Format("{0:00000}", _num1); //"00123"
       String.Format("{0:00000}",   _num2); //"00045"
       String.Format("{0:00000}", _num3); //"123456"
String.Format("{0:d5}", _num1); //"00123"
       String.Format("{0:d5}", _num2);   //"00045"
       String.Format("{0:d5}", _num3); //"123456"
```

2. n=0 için "Evet" onun dışındakiler için "Hayır" yazdıran bir deyim yazınız ?

```
Console.WriteLine(String.Format("{0:Hayır;;Evet}",  n));
```

Çıktı: n=0 ise *"Evet"*, değilse *"Hayır"* yazar.

3. Bir sayıyı string tipine dönüştüren şu kodlardan hangisi tercih edilmelidir?

```
Double testDouble = 17.73;
    // boxing yapar
String testString1 = String.Format("{0:C}", testDouble);
    // boxing yapmaz
String testString2 = testDouble.ToString("C");
```

4. Bir konsol çıktısına { } parantezleri yazdırılabilir mi?

Evet.

```
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(String.Format("{{21.yy
cahilleri}}={0}","Unutmayı ve yeniden Öğrenmeyi bilmeyenler"));
        }
    }
```

5. 03122341010 biçiminde verilen telefon numarasını 0312-234 10 10 biçiminde nasıl yazdırabilirsiniz?

```
class Program
    {
        static void Main(string[] args)
        {
            string str = String.Format("{0:0###-### ## ##}",
03122341010);
            Console.WriteLine(str);
        }
    }
```