Bölüm 16

Char Yapısı

Char Yapısı

C# dilinde `Char` bir sınıf değil, bir yapıdır (struct). `Unicode` karakterlerini temsil eder.

Yapı ile sınıf arasındaki ayrımın kullanıcı için önem taşımadığı yerlerde, `yapı` sözcüğü yerine sınıf sözcüğünü kullanacağımızı söylemiştik. Böyle yapmakla, sistematik yapıdan bir kayba uğramayız ama pedagojik açıdan biraz daha avantajlı olacağız. Çünkü başlangıçta öğrenme sürecinde fazla ayrıntıya girmemiş olacağız. Ama genellikten bir şey yitirmeyeceğiz. Çünkü, yapı ile ilgili işleri sınıf ile yapabiliriz.

Bilgisayarlar Alfabe Bilmez

İlkönce şunu bilmeliyiz. Bilgisayarlar sayıları, harfleri ve öteki simgeleri belleklerinde bizim gördüğümüz gibi tutmazlar. Onlar her şeyi ikili sayıtlama dizgesine (binary system) göre belleklerinde tutarlar. Örneğin, 'A' harfinin ASCII kodu 65 dir. Bunun ikili sayıtlama dizgesindeki karşılığı 1000001 dir. Dolayısıyla 'A' harfi için bellekte tutulan budur. Benzer olarak, '7' rakamımın ASCII kodu 55 dir. Bunun ikili sayıtlama dizgesindeki karşılığı 110111 dir. Dolayısıyla '7' rakamı için bellekte tutulan budur.

Öyleyse, bilgisayara 'A' harfini veya '7' rakamını girdi olarak verdiğimizde, sistem onu otomatik olarak ikili sistemdeki karşılığına çevirir. Bunun tersi de doğrudur. Bilgisayar belleğindeki 10000001 dizisi bize çıktı olarak gelirken 'A' simgesine dönüşür. Aslında ekranda veya printerde bize görünen 'A' çıktısı bir resimdir; 10000001 dizisinin grafiğe dönüşmüş bir temsilidir. Bu temsili latin alfabesinde, arap alfabesinde veya japon alfabesinde farklı simgelerle gösterebilirsiniz. Girdi/çıktı işlemlerinde daima bu dönüşümler

olmaktadır. Bilgisayardan karakter çıktıları bize (ekrana veya printere) birer grafik olarak geldiğine göre, çıktıyı her alfabe için istenen biçime dönüştürmek mümkündür. Bu dönüşümü işletim sistemi otomatik yapar. Programcı olarak işin o yanıyla ilgilenmeyiz. Ama o grafiklerin (karakter) çıktıda nereye yerleşeceğini ayarlayabiliriz. Genel olarak, çıktının biçemi (format) denilen bu iş, metni (string) veya sayıyı oluşturan karakterlerin istenen sıra ve biçemde yazılmasından ibarettir. Aslında string'in kendisi bir biçemdir. Bilgisayardan çıktılar bize bir string olarak ulaşır. Şimdi C# dilinde çıktının nasıl biçemlendiğini örnekler üzerinde açıklayacağız.

Unicode Sistemi

Bilgisayarın tarihi gelişimine bakarsak, karakterler önceleri 7 sonra 8 bitlik sistemle temsil edildi. 7 bit ile yazılabilecek farklı karakter sayısı 128 , 8 bit ile yazılabilecek farklı karakter sayısı 256 dır. En yaygın kullanılan ASCII kodlama sisteminde önce karakterler 7 bit ile temsil edildi. Her karaktere 0 – 127 arasında bir numara verildi. Sonra 8 bitlik genişletilmiş ASCII kodlama sistemi kullanıldı. Bir bit'in eklenerek 7 bit'lik sistemden `8 bit`'lik sisteme geçilmesi, kullanılabilecek karakter sayısını 128 den 256 ya çıkardı. Ancak, 256 karakter dünyadaki bütün alfabelere yetmedi. O nedenle, uzun süre bilgisayar dünyası Latin alfabesini kullandı. Daha sonra, piyasa talebi, farklı ülke ve kültürlere hitap etme gereğini ortaya çıkardı. Karakterleri 16 bitlik adreste tutan `unicode` sistemine geçildi. ASCII kodlama sistemindeki gibi kullanılırsa, 16 bitlik sistemde 65536 farklı karakter temsil edilebilir. Bu dünyadaki bütün alfabeler için yeterli sanılabilir. Ama yetmez. Çünkü, yalnızca Çin alfabesinde 80.000 den fazla karakter (sembol) vardır. O zaman 16 bitlik değil, 32 bitlik sistemin kullanılması bir çözüm olarak görülebilir. 32 bitlik kodlama sistemi kurulsaydı 4 294 967 295 farklı karakter temsil edilebilirdi. Ama, her karaktere 32 bitlik yer ayıran bir sistemin öğrenilmesi çok zor olacağı gibi, bellekte çok büyük yer kaplayacağı da açıktır. O nedenle, akıllıca bir yol düşünüldü. 16 bitlik sistemde, önce *taban* (*base*) olan bir grup karakter yaratıldı. Öteki karakterler bu taban karakterlerin birleşimi olarak tanımlandı. Böylece, dünyadaki bütün alfabeleri 16 bitlik sistemle temsil edebilme olanağı yaratıldı. Java, C# ve benzeri çağdaş diller `unicode` sistemini kullanır. O nedenledir ki, biz bu kitapta değişken, fonksiyon, sınıf adlandırmalarında çekinmeden ç,Ç, İ,ı, ğ,Ğ, ö,Ö, ü,Ü, ş,Ş harflerini kullanacağız. Bilindiği gibi, 8 bitlik sistemi kullanan bazı dillerde, bu harflerin ad *(identifier)* içinde kullanılması sorunlar yaratmaktadır.

Aşağıdaki program, kullanıcının klavyeden gireceği bir karakterin unicode numarasını 10 tabanlı ve 16 tabanlı sayıtlama sistemlerinde göstermektedir.

UnicodeYaz01.cs

```
using System;

namespace Methods
{
    class UnicodeYaz01
    {
        static void Main()
        {
            Console.WriteLine("Hangi harfin unicodu'nu istiyorsunuz?");
            int kod = Console.Read();
            Console.WriteLine("{0} \t {1} \t U+{2:x4}", (char)kod,
(int)kod, (int)kod);
            }
        }
    }
```

Çıktı

```
Hangi harfin unicodu'nu istiyorsunuz?
Ğ
```

```
Ğ     286    U+011e
```

Bu çıktıyı açıklamak yararlı olabilir.

Console.Read() metodu kullanıcının kalavyeden girdiği karakteri okur. Buffer'a karakterin unicode numarası binary olarak girer.

kod = Console.Read() atama deyimi kapalı (implicit) dönüşüm yapar ve buffer'daki binary değeri int tipine çevirir.

Son satırdaki {0}, {1} ve {2} yer tutucudardır. Sırasıyla, (char)kod, (int)kod, (int)kod) değişkenlerinin çıktıda yazılacağı yerleri belirlerler.

\t simgesi tab yazar, çıktıda aynı satıra yazılan ardışık değerler birbirinden bir tab kadar uzağa yerleşir.

U+{2:x4} ifadesinde U+ simgesi unicode'u gösteren bir öntakıdır, string olarak çıktıya olduğu gibi yazılır. {2:x4} simgesinde (2) yer tutucudur, 2-inci değişkenin çıktıdaki yerini belirler. (:) çıktıda biçemin başlama yeridir. (x) çıktının hexadecimal (16 tabanlı sistem) yazılmasını sağlar. (4) çıktının 4 haneye sağa yanaşık yazılmasını sağlar.

Aşağıdaki program Türk alfabesindeki büyük ve küçük harflerin karşısına `unicode` ve `hexadecimal` (16 tabanlı sayıtlama sistemi) numaralarını yazdırır.

UnicodeYaz02.cs

```
using System;

namespace Methods
{
    class UnicodeYaz02
    {
        static void Main()
        {
            string text =
"AaBbCcÇçDdEeFfGgĞğHhIıİiJjKkLlMmNnOoÖöPpRrSsŞşTtUuÜüVvYyZz";
            foreach (char c in text)
            {
                Console.Write("{0} \t {1} \t U+{2:x4}", c, (int)c,
(int)c);
                Console.WriteLine("{0} \t {1} \t U+{2:x4}", c, (int)c,
(int)c);
            }
        }
    }
}
```

Bu programın çıktısını bir tablo biçiminde gösterirsek, Türkçe alfabenin unicode numaralarını görüyor olacağız. Gerektiğinde bu tabloyu referans olarak kullanabiliriz.

Türkçe Alfabenin Unicode Kodları

Harf	Unicode	Hex	Harf	Unicode	Hex
A	65	U+0041	a	97	U+0061
B	66	U+0042	b	98	U+0062
C	67	U+0043	c	99	U+0063

Ç	199	U+00c7	ç	231	U+00e7
D	68	U+0044	d	100	U+0064
E	69	U+0045	e	101	U+0065
F	70	U+0046	f	102	U+0066
G	71	U+0047	g	103	U+0067
Ğ	286	U+011e	ğ	287	U+011f
H	72	U+0048	h	104	U+0068
I	73	U+0049	ı	305	U+0131
İ	304	U+0130	i	105	U+0069
J	74	U+004a	j	106	U+006a
K	75	U+004b	k	107	U+006b
L	76	U+004c	l	108	U+006c
M	77	U+004d	m	109	U+006d
N	78	U+004e	n	110	U+006e
O	79	U+004f	o	111	U+006f
Ö	214	U+00d6	ö	246	U+00f6
P	80	U+0050	p	112	U+0070
R	82	U+0052	r	114	U+0072
S	83	U+0053	s	115	U+0073
Ş	350	U+015e	ş	351	U+015f
T	84	U+0054	t	116	U+0074
U	85	U+0055	u	117	U+0075
Ü	220	U+00dc	ü	252	U+00fc

V	86	U+0056	v	118	U+0076
Y	89	U+0059	y	121	U+0079
Z	90	U+005a	z	122	U+007a

Bu tabloya dikkat edersek, İngiliz alfabesinde de olan harflerimiz unicode numarası olarak ASCII kodlarını almaktadır. Ama Ç, ç, Ö, ö, Ü, ü harflerinin kodları 127 bariyerini aşmaktadır, ama 255 den küçüktür. Dolayısıyla onlar genişletilmiş ASCII karakterleri sınıfındadır. Böyle olması doğaldır, çünkü bu harfler bazı avrupa ülkelerinin alfabelerinde de vardır ve genişletilmiş ASCII kümesine dahil edilmişlerdir. Öte yandan, avrupa alfabelerinde yer almayan Ğ, ğ, ı, İ, Ş, ş harflerinin unicode numaraları 255 bariyerini aşmaktadır. Onlar genişletilmiş ASCII kümesine ait değildirler.

Ayrıca şuna dikkat etmeliyiz. Alfabedeki sıralama kodlara da yansımıştır; alfabetik sıralamada önce gelen harfin kodu sonra gelen harfin kodundan daha küçüktür. Bu olgu çok önemlidir. Bilgisayarla yapılan alfabetik sıralamanın alıştığımız sırada olmasını sağlar. Çünkü alfabetik ya da daha genel olarak string sıralamaları karakterlerin unicode numaralarına göre yapılır.

Yukarıdaki programda

```
text="0123456789"
```

koyarsak, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 rakamlarının unicode ve hexadecimal kodlarını elde ederiz.

Rakam	**Unicode**	**Hex**
1	49	U+0031
2	50	U+0032
3	51	U+0033
4	52	U+0034
5	53	U+0035
6	54	U+0036
7	55	U+0037
8	56	U+0038
9	57	U+0039

Char Yapısı

Bu yapının unicode karakterlerini temsil eden bir sınıf olduğunu söylemiştik. Bu bölümde, `Char` yapısının sistematik incelemesine girmeden onun başlıca öğelerini ve ne işe yaradıklarını açıklayacağız.

Veri Alanları

İki tanedir: `MaxValue, MinValue.`

`MaxValue` temsil edilen karakter kodlarının en büyüğünü gösterir. Hexadecimal değeri 0xFFFF olan sabit sayıdır.

`MinValue` temsil edilen karakter kodlarının en küçüğünü gösterir. Hexadecimal değeri 0xFFFF olan sabit sayıdır. Hexadecimal değeri 0x00 olan sabit sayıdır.

Metotlar

Karakterlerin mukayese edilmesi, kodlarının bulunması, harf, sayı, sembol olup olmadıklarının saptanması, büyük ya da küçük harf karşılığına dönüştürülmesi, standart çıktıya string olarak yazılması gibi işleri yapan metotlar şunlardır.

CompareTo , Equals , GetHashCode , GetNumericValue , GetType , GetTypeCode , GetUnicodeCategory , IsControl , IsDigit , IsLetter , IsLetterOrDigit , IsLower , IsNumber , IsPunctuation , IsSeparator , IsSurrogate , IsSymbol , IsUpper , IsWhiteSpace , Parse , ToLower , ToString , ToUpper.

Bu metotların adları, her birinin işlevi hakkında yeterli ipucu vermektedirler. O nedenle, bir kaçıyla ilgili uygulamalar yapmakla yetineceğiz.

Char.GetNumericValue() Metodu:

Bir karakter 0,1,2,3,4,5,6,7,8,9 rakamlarından biri ise, o karakterin sayısal değerini; değilse -1 değerini alır. İki biçimi vardır:

```
public static double GetNumericValue(char c);
public static double GetNumericValue(string s,int index);
```

Örneğin,

```
GetNumericValue('3')==3
GetNumericValue('a')==-1
```

olur. Bu metot bir string içinde sıra sayısı (index) verilen bir karakter için benzer işi yapar. Örneğin,

```
GetNumericValue("abc75dfg",4))==  5
GetNumericValue("abc75dfg",6))== -1
```

olur. Çünkü birinci satırda "abc75dfg" stringindeki 4-üncü karakter 5 rakamıdır; metot 5 değerini verir. İkinci satırda 6-ıncı karakter ise rakam değil 'f' harfidir; metot -1 değerini verir.

Bu dört satırı bir program içine koyarak, çıktıları görebiliriz.

```
using System;

namespace metotlar
{
    class Char01
    {
        public static void Main()
        {
            Console.WriteLine(Char.GetNumericValue('3'));    // Çıktı:  3
            Console.WriteLine(Char.GetNumericValue('a'))   ; // Çıktı: -1
```

```
            Console.WriteLine(Char.GetNumericValue("abc75dfg", 4));  //
Çıktı:  5
            Console.WriteLine(Char.GetNumericValue("abc75dfg", 6));  //
Çıktı:  -1
        }
    }
}
```

Char.IsLetter() Metodu

Bir karakterin ya da bir string içinde sırası verilen karakterin bir harf olup olmadığını belirler. Harf ise `True` değeri, değilse `False` değeri alır.

Char02.cs

```
using System;

public class Char02
{
    public static void Main()
    {
        Console.WriteLine(Char.IsLetter('7'));              // False
        Console.WriteLine(Char.IsLetter("Başkent", 3));     // True
    }
}
```

Char.ToUpper() Metodu

Bir unicode karakteri büyük harfe dönüştürür. Tabii, dönüşecek karakterin, ilgili alfabede büyük harf karşılığının olması gerekir. Metodun iki farklı biçimi vardır. Birincisi işletim sisteminin ait olduğu dil/kültür threadinde çalışır. Diğeri ise, işletim sisteminin ait olduğu dil/kültür threadinden farklı bir thread için çalışır. Aşağıdaki örnek, her iki metodon işlevini göstermektedir.

```
public static char ToUpper(char c);
public static char ToUpper(char c,CultureInfo culture);
```

Char03.cs

```
using System;
using System.Globalization;
using System.Threading;
namespace Metotlar
{
    public class IsLetterSample
    {
        public static void Main()
        {
            Console.WriteLine(Char.ToUpper('a'));                  // A
            Console.WriteLine(Char.ToUpper('B'));                  // B
            Console.WriteLine(Char.ToUpper('w', new CultureInfo("de-
DE"))); // W
            Console.WriteLine(Char.ToUpper('W', new CultureInfo("de-
DE"))); // W
        }
    }
}
```

Aşağıdaki örnek Char sınıfının başka fonksiyonlarının kullanılışını göstermektedir.

Char04.cs

```
using System;

public class CharStructureSample
{
    public static void Main()
    {
        char chA = 'A';
        char ch1 = '1';
        string str = "test string";

        Console.WriteLine(chA.CompareTo('B'));          // Çıktı: "-1"
('A' harfi 'B' harfinden önce gelir demektir.)
        Console.WriteLine('B'.CompareTo('A'));          // Çıktı: "1"
('B' harfi 'A' harfinden önce gelir demektir.)
        Console.WriteLine(chA.Equals('A'));             // Çıktı: "True"
        Console.WriteLine(Char.GetNumericValue(ch1));   // Çıktı: "1"
        Console.WriteLine(Char.IsControl('\t'));        // Çıktı: "True"
        Console.WriteLine(Char.IsDigit(ch1));           // Çıktı: "True"
        Console.WriteLine(Char.IsLetter(','));          // Çıktı: "False"
        Console.WriteLine(Char.IsLower('u'));           // Çıktı: "True"
        Console.WriteLine(Char.IsNumber(ch1));          // Çıktı: "True"
        Console.WriteLine(Char.IsPunctuation('.'));     // Çıktı: "True"
        Console.WriteLine(Char.IsSeparator(str, 4));    // Çıktı: "True"
        Console.WriteLine(Char.IsSymbol('+'));          // Çıktı: "True"
        Console.WriteLine(Char.IsWhiteSpace(str, 4));   // Çıktı: "True"
        Console.WriteLine(Char.Parse("S"));             // Çıktı: "S"
        Console.WriteLine(Char.ToLower('M'));           // Çıktı: "m"
        Console.WriteLine('x'.ToString());              // Çıktı: "x"
    }
}
```

```
using System;

public class CharDemo
{
    public static void Main()
    {
        string str = "This is a test. $23";
        int i;

        for (i = 0; i < str.Length; i++)
        {
            Console.Write(str[i] + " is");
            if (Char.IsDigit(str[i]))
                Console.Write(" digit");
            if (Char.IsLetter(str[i]))
                Console.Write(" letter");
            if (Char.IsLower(str[i]))
                Console.Write(" lowercase");
            if (Char.IsUpper(str[i]))
                Console.Write(" uppercase");
            if (Char.IsSymbol(str[i]))
                Console.Write(" symbol");
            if (Char.IsSeparator(str[i]))
```

```
                Console.Write(" separator");
            if (Char.IsWhiteSpace(str[i]))
                Console.Write(" whitespace");
            if (Char.IsPunctuation(str[i]))
                Console.Write(" punctuation");

            Console.WriteLine();
        }

        Console.WriteLine("Original: " + str);

        // Convert to upper case.
        string newstr = "";
        for (i = 0; i < str.Length; i++)
            newstr += Char.ToUpper(str[i]);

        Console.WriteLine("Uppercased: " + newstr);
    }
}
```

Alıştırmalar

1. İki stringin ilk harflerinin aynı olup olmadığını bulan aşağıdaki programı çözümleyiniz. Programı değiştirerek, stringleri kullanıcının girmesini sağlayınız.

```
using System;

class Test
{
    string s1 = "Merhaba";
    string s2 = "Dünya";
    static void Main(string[] args)
    {
        Test t = new Test();
        if (t.s1[0] == t.s2[0])
            Console.WriteLine("İlk harfler aynıdır");
        else
            Console.WriteLine("İlk harfler farklıdır");
    }
}
```

2. Aşağıdaki program bir tümcedeki sözcüklerin sırasını ters çevirmektedir. Programı çözümleyiniz.

```
using System;
public static class TersKelimeler
{
    public static string TersKelime(this string str, char ayrac)
    {
        char yedek;
        int sol = 0, orta = 0;

        char[] chars = str.ToCharArray();
```

```
        Array.Reverse(chars);

        for (int i = 0; i <= chars.Length; i++)
        {
            if (i != chars.Length && chars[i] != ayrac)
                continue;

            if (sol == i || sol + 1 == i)
            {
                sol = i + 1;
                continue;
            }

            orta = (i - sol - 1) / 2 + sol;

            for (int j = i - 1; j > orta; j--, sol++)
            {
                yedek = chars[sol];
                chars[sol] = chars[j];
                chars[j] = yedek;
            }

            sol = i + 1;
        }

        return new String(chars);
    }

    public static string TersKelime(this string str)
    { return str.TersKelime(' ');
    }
    public static void Main()
    {
        string str = "Yüzyılın cahilleri kimler?";

        string strTers = TersKelimeler.TersKelime(str);
        Console.WriteLine(str);
        Console.WriteLine(strTers);
    }
}
```

3. Bir stringdeki karakterlerin sırasını ters çeviren bir program yazınız.