**Temel Veri Tipleri ve CTS(Common Type System) Karşılıkları**

C# da iki çeşit veri tipi vardır:

* Değer Tipleri (Value type)
* Referans Tipleri(Reference Type)

Değişkenler bellekte bulunan verilerdir. Bir değişkeni kullandığımız zaman o değişkenin bellekte bulundu yerdeki bilgiyi kullanırız. Değer tipleri veriyi direk olarak bellek bölgesinden alırken referans tipleri başka bir nesneye referans gösterirler. Yani referans tipleri içinde veri değil adres bilgisi tutarlar. Veri tipleri stack dediğimiz bellek bölgesinde tutulurlar, referans tipleri ise heap bellek bölgesinde saklanırlar. int, double, float gibi veri tipleri değer tiplerine örnek gösterilebilir. Herhangi bir sınıf türü ise referans tipine örnek gösterilebilir. Değer tipleri birbirine eşitlenirken değişkenin barındırdığı değer bir diğer değişkene kopyalanır. Böylece iki farklı bağımsız değişken oluşur. Referans tipleri ise eşitleme sırasında değişkenlerin taşıdıkları veri değil heap bölgesinde işaret ettikleri adres kopyalanır. Böylece eğer iki referans değişkeni birbirine eşitledi isek ve daha sonra bunlardan birinde bulunan veriyi değiştirdi ise otomatik olarak diğer referans değişkeninin değeri de değişir. Çünkü adresde bulunan veri değişince bu adresi işaret eden iki değişkende yeni veri bilgisine ulaşır.

Bir önceki dersimizde Common Type System hakkında bilgi vermiştik. C# da tanımlanan veri tipleri ve bunların CTS deki karşılıklarını tablo halinde aşağıda bulabilirsiniz. C# da toplam 15 veri tipi vardır bunlardan 13 ü değer tipindedir, 2 si ise referans tipindedir.

## Değer Tipleri:

|  |  |  |  |
| --- | --- | --- | --- |
| **C# Adı** | **CTS Karşılığı** | **Açıklama** | **Max ve Min aralık yada değeri** |
| sbyte | System.Byte | 8 bit işaretli tamsayı | -128 : 127 |
| short | System.Int16 | 16 bit işaretli tamsayı | -32.768 : 32.767 |
| int | System.Int32 | 32 bit işaretli tamsayı | -2.147.483.648 : 2.147.483.647 |
| long | System.Int64 | 64 bit işaretli tamsayı | -9.223.372.036.854.775.808 : -9.223.372.036.854.775.807 |
| byte | System.Byte | 8 bit işaretsiz tamsayı | 0,177083333 |
| ushort | System.UInt16 | 16 bit işaretsiz tamsayı | 0 : 65.535 |
| uint | System.UInt32 | 32 bit işaretsiz tamsayı | 0 : 4.294.967.295 |
| ulong | System.UInt64 | 64 bit işaretsiz tamsayı | 0 : 18.446.744.073.709.551.615 |
| float | System.Single | 32 bit tek kayan sayı | +yada - 1,5\*10-45 : + ya da - 3,4\*1038 |
| double | Sytem.Double | 64 bit çift kayan sayı | +yada - 5\*10-324 : + ya da - 1,7\*10308 |
| decimal | System.Decimal | 128 bit ondalıklı sayı | +yada - 1,5\*10-28 : + ya da - 7,9\*1028 |
| bool | System.Boolean |  | true ya da false |
| char | System.Char | Karakterleri temsil eder | 16 Unicode karakterleri |

## Referans Tipleri:

|  |  |  |
| --- | --- | --- |
| **C# Adı** | **CTS Karşılığı** | **Açıklama** |
| object | System.Object | Bütün veri türlerinin türediği kök eleman |
| string | System.String | Unicode karakterlerinden oluşan string |

## Değişkenler:

Bir program çalıştırırken tüm veriler geçici olarak hafızada tutulur. Değişkenler bu verilerin isimlendirilmesi işlemidir. C# da değişkenler aşağıdaki yapıda tanımlanırlar.

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1 | <veri tipi><değişken adı>; |

### Örneğin;

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1 | int i; |

Yukarıdaki ifade bir integer veri tipini saklamak için programın diğer aşamalarında i ismi ile ifade edilecek hafızada 4 bytelık bir alan açar. Bir değişkene tanımlarken değer atayabilirsiniz, ayrıca bir veri tipi altında birden fazla isimle farklı değişkenler tanımlayabilirsiniz.

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2  3 | bool dogruMu = false;  double yuzde = 98.32, ortalama = 35.32;  char karakter = 'A'; |

Diğer modern programlama dilleri gibi C# dada değişkenleri kullanmadan önce tanımlamak zorundasınız. Ayrıca eğer bir yerel değişken (Bir method içinde tanımlanan değişkenler) tanımlıyorsanız bu değişkeni kullanmadan önce bir değer ataması yapmak zorundasınız.

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | static void Main(string[] args)  {        int kilo;        //kilo = 78;        Console.WriteLine(kilo);  } |

Yukarıdaki kod parçacığı kilo değişkenine değer atamadığınız için hata verecektir çünkü kilo değişkeni yerel bir değişkendir. Eğer //kilo = 78; açıklama satırındaki // işaretlerini siler ve bu satırı koda dahil ederseniz hatanın ortadan kalktığını görürsünüz. Buradan da anlaşılacağı gibi C# yerel değişkenlere ilk atama işlemini otomatik yapmak global değişkenler de ise ilk değer ataması değişken tipine göre otomatik olarak yapılır. Örneğin integer bir tip için 0 gibi.

## Sabit Değişkenler:

Sabit değişkenler içinde buluna veri değişken ömrü tamamlanana kadar değişmez. Bir sabit tanımlamak için**const** anahtar kelimesini kullanırız. İlk tanımlandıkları sırada ilk değer ataması yapmak zorundayız, uygulamamızın herhangi bir kısmında sabit değişkenin içeriğini değiştiremeyiz.

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2 | const double PI = 3.14;  const double PI; |

Yukarıdaki tanımlamalardan birincisi doğru ikincisi ise yanlış bir tanımlamadır. İlk atamaları yapılmak zorundadır.

Sabitler genellikle büyük harflerle tanımlanırlar. Bu bir zorunluluk değildir.

## Değişkenler ve Metotlar için İsimlendirme Konvansiyonları

Microsoft değişken isimlendirmesi için **Camel Notasyonu** methotlar için ise **Pascal Notasyonunu**tavsiye ediyor.

Camel notasyonunda isim küçük harfle başlar, eğer değişken isminde birden fazla kelime geçiyor ise isimdeki diğer kelimeler büyük harfle başlar.

### Camel Notasyonu:

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2  3 | maas;  maasMiktari;  massMiktariAciklama; |

Pascal Notasyonunda kelime büyük harfle başlar. Camel Notasyonunda olduğu gibi diğer kelimelerde büyük harfle başlar.

### Pascal Notasyonu:

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2 | Maas();  MaasHesapla(); |

Bu notasyonların kullanımı mecburi değildir. Fakat sürekli olarak bu tür bir notasyona uyarak kodlarınızı yazarsanız, kodlarınız daha anlaşılır bir hale girer.

## C# da Operatörler

### Aritmetik Operatörler

|  |  |
| --- | --- |
| **Operatör** | **Açıklama** |
| + | Ekleme |
| - | Çıkarma |
| \* | Çarpma |
| / | Bölme |
| % | Kalan veya modül |
| ++ | Birer Birer Arttırma |
| -- | Birer Birer Eksiltme |

Yukarıdaki tablodaki operatörlerin kullanımı ile ilgili örnek kod aşağıdadır. Ayrıca kodu [buradan](http://www.yazilimmutfagi.com/10184/Makale/Dersler/CSharp/Ders2/AritmetikOperatorler.rar) indirerek kendiniz deneyebilirsiniz.

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | public static void Main()  {        int toplam = 0, fark = 0, carpim = 0, kalan = 0;      float bolum = 0;      int sayi1 = 0, sayi2 = 0;        Console.WriteLine("Sayı Biri Giriniz : ");      sayi1 = Convert.ToInt32(Console.ReadLine());        Console.WriteLine("Sayı İkiyi Giriniz : ");      sayi2 = Convert.ToInt32(Console.ReadLine());        toplam = sayi1 + sayi2;      fark = sayi1 - sayi2;      carpim = sayi1 \* sayi2;      kalan = sayi1 % sayi2;      bolum = sayi1 / sayi2;        Console.WriteLine("Girilen Sayılar: Sayı 1 = {0}, Sayı 2 = {1}",              sayi1, sayi2);      Console.WriteLine("Sayıların Toplamı = {0}", toplam);      Console.WriteLine("Sayıların Farkı (sayi1 - sayi2) = {0}", fark);      Console.WriteLine("Sayıların Çarpımı = {0}", carpim);      Console.WriteLine("Sayıların kalan (sayi1 in sayi2 ye bölümğnden kalan)= {0}"              , kalan);      Console.WriteLine("Sayıların Bölümünden (sayi1 / sayi2) Bölüm = {0}", bolum);        sayi1++;      sayi2--;      Console.WriteLine("Sayi 1 in bir fazlası = {0}", sayi1);      Console.WriteLine("Sayi 2 in bir eksiği = {0}", sayi2);  } |

Yukarıdaki kodu inceleyecek olursanız. **Console.WriteLine("Açıklama Metni {0}" , sayi1);** gibi bir yapı görürsünüz. String ifadelerde stringİfade1 + stringİfade2 şeklinde birleştirme yapmak yerine bu formatta stringleri birleştirebilirsiniz. {0}  şeklindeki ifade virgülden sonra yazılan ilk değişkenin değerini buraya yaz demek oluyor. Eğer ikinci bir değişken varsa {1} şeklinde yazarak onunda değerini başka bir kısma yerleştirebiliriz. Dikkat etmeniz gereken {} ifadesinin içindeki sayılar sırası ile gitmeli. Yani 0, 1, 2,..

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1 | sayi1++ ifadesi |

ile

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1 | sayi1 = sayi1 + 1; |

aynı anlama gelmektedir. Ayrıca bir sayıya herhangi bir sayı eklemek istiyorsak örneğin sayi1 e 5 ekleyelim.

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1 | sayi1 = sayi1 +5 |

yazmak yerine;

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1 | sayi += 5; |

daha kısa ve doğrudur.

### Ön Ek ve Arka Ek Olarak Kullanma

++ ve – operatörleri en ve arka ek olarak kullanılabilir. Örneklerle anlatacak olursak;

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2  3 | sayi2 = 3;  sayi1 = ++sayi2;//sayi1 = 4, sayi2= 4 olur.  sayi1 = --sayi2;//sayi1 = 2, sayi2= 2 olur. |

Örnekte görüldüğü gibi eğer operator ön ek olarak kullanılırsa derleyici önce operatörün görevini yapar daha sonra ise atama işlemini gerçekleştirir. Böylece sayi1 = ++sayi2 ifadesinde sayi2 önce bir arttırılır daha sonra sayi1’e sayi2 nin değeri atanır.

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2  3 | sayi2 = 3;  sayi1 = sayi2++; //sayi1 = 3, sayi2= 4 olur.  sayi1 = sayi2--; //sayi1 = 3, sayi2= 2 olur. |

Operatör arka ek olarak kullanıldığında ise önce atama işlemi yapılar daha sonra operatörün görevi yerine getirilir. sayi1 = sayi2++ işleminde önce sayi2 nin değeri sayi1’e atanır, daha sonra sayi2 bir arttırılır.

### Atama Operatörleri

Atama Operatörleri değişkenlere değer atamak için kullanılırlar.

|  |  |
| --- | --- |
| **Operatör** | **Açıklama** |
| = | Basit eşitleme |
| += | Sağdaki Sayı ile topla sonra toplamı eşitle |
| -= | Sağdaki Sayı yı çıkar sonra sonucu eşitle |
| \*= | Sağdaki Sayı ile çarp sonra toplamı eşitle |
| /= | Sağdaki Sayı ile böl sonra bölümü eşitle |
| %= | Sağdaki Sayı ile bölümünden kalanı bul sonra kalanı eşitle |

### İlişkisel(Karşılaştırma) Operatörleri

İlişkisel Operatörler genelde koşul ifadelerinde karşılaştırma için kullanılırlar. C#da kullanılan ilişkisel operatörler aşağıdakilerdir.

|  |  |
| --- | --- |
| Operatör | Açıklama |
| == | Eşittir |
| != | Eşit Değildir |
| > | Büyüktür |
| < | Küçüktür |
| >= | Büyük Eşittir |
| <= | Küçük Eşittir |

İlişkisel Operatörler her zaman true veya false olmak üzere boolean bir değer döndürürler.  Örnekle anlatmamız gerekirse;

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | int sayi1 = 6, sayi2 = 4;  sayi1 == sayi2 //false;  sayi1 != sayi2 //true;  sayi1 > sayi2 //true;  sayi1 < sayi2 //false;  sayi1 >= sayi2 //true;  sayi1 <= sayi2 //false; |

İlişkisel operatörler sadece birbiri ile uyumlu tipteki değişkenleri karşılaştırabilir. Bir boolean tipi ile integer tipindeki iki değişkeni karşılaştıramazsınız.

### Mantıksal ve Bit Düzeyinde (Logical and Bitwise) Opeatörleri

Bu operatörler mantıksal işlemleri ve bit düzeyindeki işlemleri yapmak için kullanılırlar.

|  |  |
| --- | --- |
| **Operatör** | **Açıklama** |
| & | Bit bazında **ve**işlemi |
| | | Bit bazında **or** işlemi |
| ^ | Bit bazında **xor** işlemi |
| ! | Bit bazında **not** işlemi |
| && | Mantıksal **ve** işlemi |
| || | Mantıksal **or** işlemi |

Bit bazındaki operatörler değişkene ait değeri bit bazında işleme tabi tutarlar. Mantıksal operatörler ise bildiğimiz mantık kavramı kapsamında bu ve şu doğrumu veya bu veya şu doğrumu gibi karşılaştırma yaparlar.

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1 | bool sonuc = (i > 3 && j < 5) || (i < 7 && j > 3); |

**Mantıksal operatörlerde;**

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Doğru ve Doğru = Doğru;  Doğru ve Yanlış = Yanlış;  Yanlış ve Doğru = Yanlış;  Yanlış ve Yanlış = Yanlış;  Doğru veya Doğru = Doğru;  Doğru veya Yanlış = Doğru;  Yanlış veya Doğru = Doğru;  Yanlış veya Yanlış = Yanlış; |

Mantıksal And işlemlerinde eğer koşullardan herhangi biri yanlış ise and işlemine tabi tutulan diğer koşullar true olsa bile sonuç falsedur. Or işleminde ise koşullardan herhangi birinin true olması diğer koşulların ne olduğuna bakmaksızın sonucun true olmasını sağlar.

C# Mantıksal operatörler Kısa Devre mantığıyla çalışır. Mantıksal işlemlerde koşullar soldan sağa doğru kontrol edilir. Yukarıdaki paragraftaki bilgiye göre; bir and işlemi kontrol ediliyorsa ve koşullardan birisi false ise sağda kaç adet daha and koşulu olmasına bakmaksızın sonucu false olarak belirleyebiliriz. C# derleyicisi de bu yöntemi kullanarak and işleminde ilk false gördüğü anda or işleminde ise ilk true gördüğü anda işlemi keser ve sonucu döndürür. Bu büyük uygulamalarda çok fazla koşul olduğu zamanlar uygulamamızın daha hızlı çalışmasını sağlar.

### C# da Kullanılan Diğer Operatörler

|  |  |
| --- | --- |
| **Operatör** | **Açıklama** |
| >> | Bit bazında sağa kaydırma |
| << | Bit bazında sola kaydırma |
| , | Nesnelerin özelliklerine ulaşmak için |
| [] | İndeks numarası ile dizi ve kolleksiyonların elemanlarına ulaşmak |
| () | Çevrim Operatörü. Tip çevrimleri için kullanılır. |
| ?: | Koşul Operatörü. if else koşulunun kısa yazımı. İlerde anlatılacak |

### Operatörlerde Çalışma Önceliği

Tüm operatörlerin çalışma önceliği aynı değildir. Matematik kurallarında geçerli olan çarpma, bölme, çıkarma toplama işlemlerindeki öncelik sırası operatörler içinde geçerlidir. Öncelikleri belirlemek için işlemleri parantez içinde yazmamız gerekir. Örneğin

[?](http://www.yazilimmutfagi.com/10184/dersler/MakaleOku.aspx?Makale=YazIlIm+MutfagI+Dersleri+(CSharp)+Ders+2+-+Degisken+Tipleri,+Degisken+TanImlama+ve+Operatorler&MakaleKey=13e2a215-a5a0-4716-a3bf-91a7393b551e)

|  |  |
| --- | --- |
| 1  2  3  4  5 | int i = 3 + 2 \* 6;  // önce 2 ile 6 çarpılır daha sonra çarpıma 3 eklenerek 15 bulunur.    int j = (3 + 2) \* 6;  // önce 3 ile 2 toplanır daha sonra toplam 6 ile çarpılarak 30 bulunur. |