OPERATÖRLER
Alıştırmalar

Aritmetik Operatörleri

Operatör	*Açıklama*	*Grup*
+	Toplama, artı işleci	İkili İşlem (binary operator)
-	Çıkarma, eksi işleci	İkili İşlem (binary operator)
*	Çarpma, çarpım işleci	İkili İşlem (binary operator)
/	Bölme, bölüm işleci	İkili İşlem (binary operator)
%	Modulo, kalan işleci	İkili İşlem (binary operator)
++	Artım (auto increment)	Birli İşlem (unary operator)
--	Eksim (auto decrement)	Birli İşlem (unary operator)

Örnek:

```
using System;

class Binary
{
    public static void Main()
    {
        int x, y, sonuç;
        float floatsonuç;

        x = 7;
        y = 5;

        sonuç = x + y;
        Console.WriteLine("x+y: {0}", sonuç);

        sonuç = x - y;
        Console.WriteLine("x-y: {0}", sonuç);

        sonuç = x * y;
        Console.WriteLine("x*y: {0}", sonuç);

        sonuç = x / y;
        Console.WriteLine("x/y: {0}", sonuç);

        floatsonuç = (float)x / (float)y;
```

```
        Console.WriteLine("x/y: {0}", floatsonuç);

        sonuç = x % y;
        Console.WriteLine("x%y: {0}", sonuç);

        sonuç += x;
        Console.WriteLine("sonuç+=x: {0}", sonuç);
    }
}
```

Çıktı:

```
x+y: 12
x-y: 2
x*y: 35
x/y: 1
x/y: 1.4
x%y: 2
sonuç+=x: 9
```

Örnek:

```
using System;
namespace Operator
{
    class Aritmetik
    {
        static void Main(string[] args)
        {
            int x = 10;
            int y = 5;
            int sonuç;

            sonuç = x + y;
            Console.WriteLine("x + y = {0}", sonuç);

            sonuç = x - y;
            Console.WriteLine("x - y = {0}", sonuç);

            sonuç = x * y;
            Console.WriteLine("x * y = {0}", sonuç);

            sonuç = x / y;
            Console.WriteLine("x / y = {0}", sonuç);
        }
    }
}
```

Çıktı:

```
x + y = 15
x - y = 5
x * y = 50
x / y = 2
```

Yukarıdaki işlemleri **sonuç** değişkenini hiç kullanmadan yapabiliriz:

Örnek:

```
using System;
namespace Operator
{
    class Aritmetik
    {
        static void Main(string[] args)
        {
            int x = 10;
            int y = 5;
            int sonuç;
            Console.WriteLine("x + y = {0}", x + y);

            Console.WriteLine("x - y = {0}", x - y);

            Console.WriteLine("x * y = {0}", x * y);

            Console.WriteLine("x / y = {0}", x / y);
        }
    }
}
```

Örnek:

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

        int a = 2;
        int b = 6;

        int c = 12;
        int d = 5;

        Console.WriteLine(b % a);
        Console.WriteLine(c % d);
        Console.Read();
    }
}
```

Çıktı

```
0
2
```

Örnek:

```
using System;

namespace operators
{
    class Aritmetik03
    {
        static void Main(string[] args)
        {
            double x = 10.34;
            double y = 4;
            double result = x % y;
            Console.WriteLine("x % y = {0}", result);

        }
    }
}
```

Çıktı:

```
x % y = 2,34
```

Tamsayı Bölme

/ bölme işlemi, aksi istenmediği zaman bölümün tamsayı kısmını verir. Eğer bölümün kesir kısmını istiyorsak, bunu istemli dönüşüm (explicit casting) ile belirtmeliyiz.

Örnek:

```
using System;

class Binary
{
    public static void Main()
    {
        int x, y, sonuç;
        float floatsonuç;

        x = 7;
        y = 5;

        sonuç = x + y;
```

```
        Console.WriteLine("x+y: {0}", sonuç);

        sonuç = x - y;
        Console.WriteLine("x-y: {0}", sonuç);

        sonuç = x * y;
        Console.WriteLine("x*y: {0}", sonuç);

        sonuç = x / y;
        Console.WriteLine("x/y: {0}", sonuç);

        floatsonuç = (float)x / (float)y;
        Console.WriteLine("x/y: {0}", floatsonuç);

        sonuç = x % y;
        Console.WriteLine("x%y: {0}", sonuç);

        sonuç += x;
        Console.WriteLine("sonuç+=x: {0}", sonuç);
    }
}
```

Çıktı:

```
x+y: 12
x-y: 2
x*y: 35
x/y: 1
x/y: 1,4
x%y: 2
sonuç+=x: 9
Devam etmek için bir tuşa basın . . .
```

int, float ve double tiplerinin işleme girişleri:

Örnek:

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

        int a = 2;
        int b = 6;
        int c = a - b;

        float d = 1;
        float e = 11.1f;
        double f = 1;

        Console.WriteLine(c);
```

```
        Console.WriteLine(d + e);
        Console.WriteLine(f + e);
    }
```

Çıktı

```
-4
12,1
12,1000003814697
```

Önel ve Sonal Artım ve Eksim (++, --)

Örnek:

```
class Program
{
    static void Main(string[] args)
    {

        int y;
        int a = 100;

        y = ++a;   //önel-artım (pre-increment)
        Console.WriteLine("y = " + y + " and a = " + a);
        //y=101  and a=101

        y = a++;   //sonal-artım (post-increment)
        Console.WriteLine("y = " + y + " and a = " + a);
        //y=101  and a=102
    }
}
```

Çıktı:

```
y = 101 and a = 101
y = 101 and a = 102
```

Örnek:

```
using System;

class Unary
{
    public static void Main()
    {
```

```
        int unary = 0;
        int preIncrement;
        int preDecrement;
        int postIncrement;
        int postDecrement;
        int positive;
        int negative;
        sbyte bitNot;
        bool logNot;

        preIncrement = ++unary;
        Console.WriteLine("pre-Increment: {0}", preIncrement);

        preDecrement = --unary;
        Console.WriteLine("pre-Decrement: {0}", preDecrement);

        postDecrement = unary--;
        Console.WriteLine("Post-Decrement: {0}", postDecrement);

        postIncrement = unary++;
        Console.WriteLine("Post-Increment: {0}", postIncrement);

        Console.WriteLine("Final Value of Unary: {0}", unary);

        positive = -postIncrement;
        Console.WriteLine("Positive: {0}", positive);

        negative = +postIncrement;
        Console.WriteLine("Negative: {0}", negative);

        bitNot = 0;
        bitNot = (sbyte)(~bitNot);
        Console.WriteLine("Bitwise Not: {0}", bitNot);

        logNot = false;
        logNot = !logNot;
        Console.WriteLine("Logical Not: {0}", logNot);
    }
}
```

Çıktı:

```
pre-Increment: 1
pre-Decrement: 0
Post-Decrement: 0
Post-Increment: -1
Final Value of Unary: 0
Positive: 1
Negative: -1
Bitwise Not: -1
Logical Not: True
```

+ ve - operatörleri

+ ve – operatörleri **birli (unary)** operatörlerdir. + operatörü, önüne geldiği sayının işaretini değiştirmezken, - operatörü önüne geldiği sayının işaratini değiştirir; yani artı sayıyı eksi, eksi sayıyı artı yapar. Aşağıdaki örnek bunun nasıl olduğunu göstermektedir.

Örnek:

```
using System;
namespace Operators
{
    class Class1
    {
        static void Main(string[] args)
        {
            int x = -10;
            int y = +10;
            int z = +-10;
            int a = 3;
            int sonuç;
            Console.WriteLine("x = {0}, y = {1}, z = {2}", x, y, z);

             z = -10;
             sonuç = z * a;
            Console.WriteLine("sonuç = {0}", sonuç);
             sonuç = z * +a;
            Console.WriteLine("sonuç = {0}", sonuç);
            Console.ReadLine();
        }
    }
}
```

Çıktı:

```
x = -10, y = 10, z = -10
sonuç = -30
sonuç = -30
```

Atama (assignment) Operatörleri

Atama operatörleri, değişkenlere değer atamak için kullanılan işleçlerdir. C# dilinde aşağıdaki atama operatörleri kullanılır:

Operatör	Açıklama
=	Sağdaki değeri soldaki değişkene atar.
+=	Soldakine sağdakini ekler, sonucu soldakine atar.
-=	Soldakinden sağdakini çıkarır, sonucu soldakine atar.
*=	Soldakini sağdaki ile çarpar, sonucu soldakine atar.
/=	Soldakini sağdakine böler, sonucu soldakine atar.
%=	Soldaki ile sağdakinin modula işleminin sonucunu soldakine atar.

Örnek:

```
using System;
namespace Operators
{
    class Class1
    {
        static void Main(string[] args)
        {
            int x = 8;
            int y = 6;
          Console.WriteLine(" Başlangıçta x = {0} ve  y = {1} dır.", x, y);

          Console.WriteLine("**********");

             x = x + y;
          Console.WriteLine("x = x + y atamasından sonra  x = {0}  olur", x);
             x = 8;  x += y;
          Console.WriteLine("x += y    atamasından sonra  x = {0}  olur", x);

          Console.WriteLine("**********");

             x = 8;  x = x - y;
          Console.WriteLine("x = x - y  atamasından sonra  x = {0}  olur", x);
             x = 8; x -= y;
          Console.WriteLine("x -= y     atamasından sonra  x = {0}  olur", x);

          Console.WriteLine("**********");

             x = 8; x = x * y;
          Console.WriteLine("x = x * y  atamasından sonra  x = {0}  olur", x);
             x = 8; x *= y;
          Console.WriteLine("x *= y     atamasından sonra  x = {0}  olur", x);

          Console.WriteLine("**********");

             x = 8; x = x / y;
```

```
            Console.WriteLine("x = x / y  atamasından sonra  x = {0}  olur", x);
               x = 8; x /= y;
            Console.WriteLine("x /= y     atamasından sonra  x = {0}  olur", x);

            Console.WriteLine("**********");

               x = 8; x = x % y;
            Console.WriteLine("x = x % y  atamasından sonra  x = {0}  olur", x);
               x = 8; x %= y;
            Console.WriteLine("x %= y     atamasından sonra  x = {0}  olur", x);
         }
    }
}
```

Çıktı:

```
Başlangıçta x = 8 ve  y = 6 dır.
**********
x = x + y atamasından sonra  x = 14  olur
x += y    atamasından sonra  x = 14  olur
**********
x = x - y  atamasından sonra  x = 2  olur
x -= y     atamasından sonra  x = 2  olur
**********
x = x * y  atamasından sonra  x = 48  olur
x *= y     atamasından sonra  x = 48  olur
**********
x = x / y  atamasından sonra  x = 1  olur
x /= y     atamasından sonra  x = 1  olur
**********
x = x % y  atamasından sonra  x = 2  olur
x %= y     atamasından sonra  x = 2  olur
```

İlişkisel Operatörler

İlişkisel (relational) operatörler iki değişkenin değerlerini karşılaştırır. Dolayısıyla, karşılaştırılan öğeler arasında bir boolean deyim kurar. Karşılaştırılan değerlerin eşitliğini ya da birinin ötekinden büyük ya da küçük olduğunu belirten boolean deyim `doğru` ya da `yanlış` (`true, false`) değerlerden birisini alır. C# dilinde aşağıdaki ilişkisel operatörler kullanılır.

Operatör	Açıklama
==	Eşit mi?
!=	Eşit-değil mi?
>	Büyük mü?
<	Küçük mü?
>=	Büyük veya eşit mi?
<=	Küçük veya eşit mi?

Mantıksal (Logic) Operatörler

C# dilinde, mantıksal deyimleri birbirlerine bağlamak için iki operatör kullanılır

```
&&   Logical  AND     (Mantıksal VE)
||   Logical  OR      (Mantıksal VEYA)
```

&& Operatörü

boolean1 ve boolean2 iki mantıksal deyim olmak üzere, bu iki deyimin mantıksal VE ile birbirlerine bağlanması için

```
boolean1 && boolean2
```

Bitsel (Bitwise) Operatörler

Bitsel işlemler yapmak için kullanılan operatörledir.

Operatör	Açıklama
&	bitsel AND
|	bitsel OR
^	bitsel XOR
!	**bitsel NOT**

Başka Operatörler

C# dilinde aşağıdaki operatörleri de kullanırız.

Operand	Açıklama	
<<	Sola kayan bit işlemi	(left shift bitwise operator)
>>	Sağa kayan bit işlemi	(right shift bitwise operator)
.	Nesne öğelerine erişim	(member access for objects)
[ ]	Array indisleme	(indexing operator used in arrays and collections)
( )	Veri dönüştürme operatörü	(cast operator)
? :	Koşullu deyim operatörü	(ternary operator)