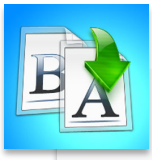
OPERATÖRLER



İÇİNDEKİLER

- Operatör Nedir?
- Aritmetik Operatörler
- Atama Operatörleri
- Mantıksal Operatörler
- Karşılaştırma Operatörleri
- Bitsel Operatörler

HEDEFLER

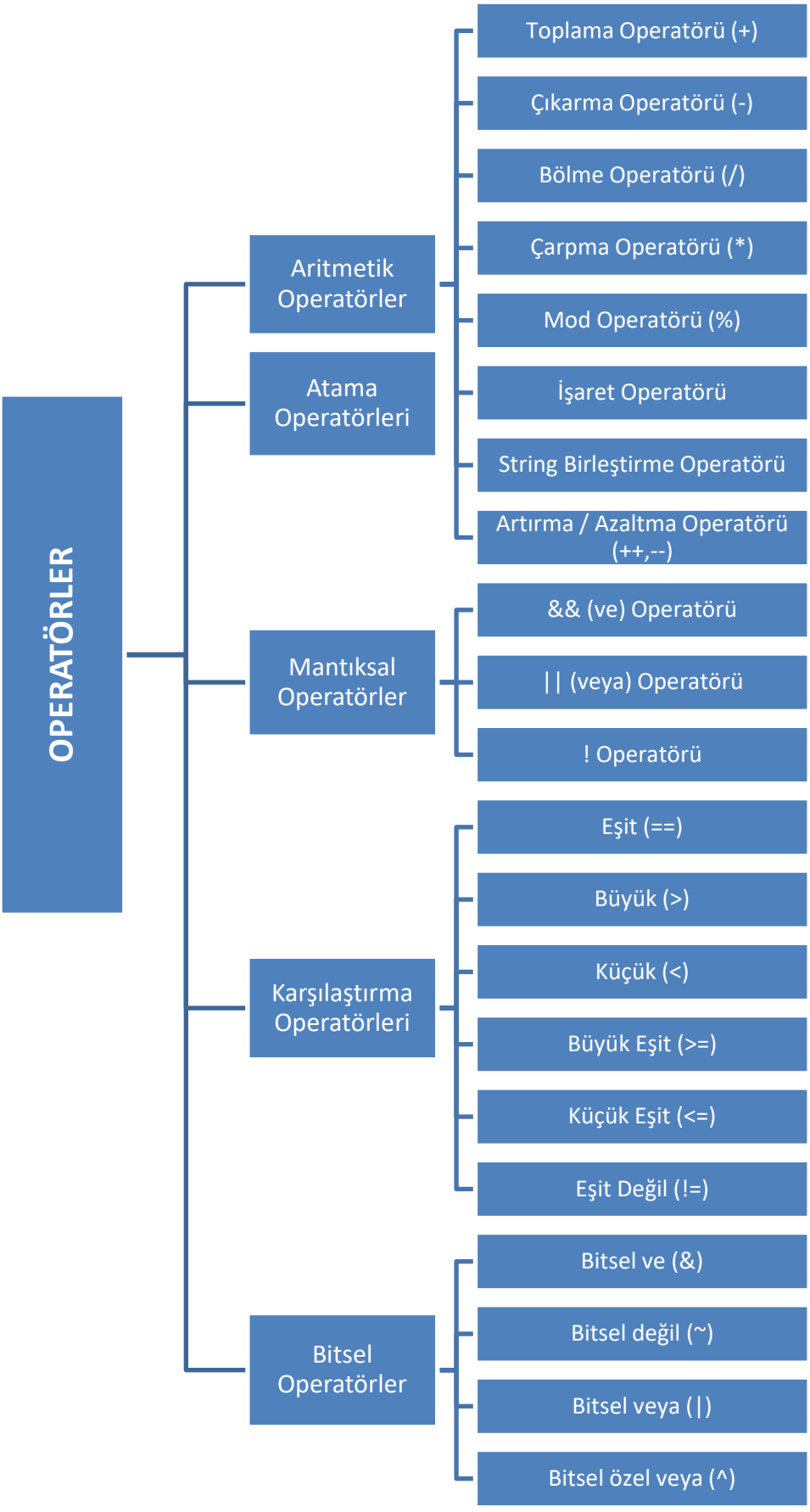
- Bu üniteyi çalıştıktan sonra;
  - Operatörleri tanıyabilecek,
  - Operatörleri sınıflandırabilecek,
  - Operatörleri kullanarak çeşitli işlemler yapabilecek,
  - Operatörlerin kullanım mantığını anlayabileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Öğr. Gör.
Çağrı Burak ARINÇ

ÜNİTE
7


OPERATÖRLER
Aritmetik Operatörler
Atama Operatörleri
Mantıksal Operatörler
Karşılaştırma Operatörleri
Bitsel Operatörler
Toplama Operatörü (+)
Çıkarma Operatörü (-)
Bölme Operatörü (/)
Çarpma Operatörü (*)
Mod Operatörü (%)
İşaret Operatörü
String Birleştirme Operatörü
Artırma / Azaltma Operatörü (++,--)
&& (ve) Operatörü
|| (veya) Operatörü
! Operatörü
Eşit (==)
Büyük (>)
Küçük (<)
Büyük Eşit (>=)
Küçük Eşit (<=)
Eşit Değil (!=)
Bitsel ve (&)
Bitsel değil (~)
Bitsel veya (|)
Bitsel özel veya (^)

GİRİŞ

Dil insan hayatının vaz geçilemez en temel ögelerinden biridir. Çünkü bireyler arasındaki iletişimin ve temel ihtiyaçların giderilmesinin aracı dildir. Ortak bir dil anlayışının gelişmesinde Sümerler tarafından MÖ 3000'li yıllarda yazının keşfedilmesinin büyük rolü vardır. Günlük hayatımızda iletişim kurabilmek için cümle yapısını kullanırız. Kullandığımız her cümle edatlar ve bağlaçlar gibi birçok yapı ile birbirine bağlanarak anlam kazanır. Programlama dillerinde de bu yapıya benzer ifadeler bulunmaktadır. Programlama dilinde ise sesler ve sözcüklerin yerini kodlar veya semboller almıştır. Bu yapılar programlama dillerinde "operatör" ismini almaktadır. Operatörler; hesaplamalar, mantıksal işlemler ya da kontrol fonksiyonları gibi birçok yerde kullanılmaktadır. Programlama dilleri kullanıcının yapmak istediği işlemleri genellikle operatör sayesinde anlamlandırır ve işlem yapabilme kabiliyetine sahip olur.

Programlama dilleri incelendiğinde operatörlerin her dilde aynı yapıya sahip olmadığı görülmektedir. Fakat kullanılan bazı temel operatörler programlama dillerinin hemen hemen hepsinde aynıdır. Örneğin toplama ve çıkarma işlemi, bütün programlama dillerinde bulunmaktadır. Bu bölümde C# programlama dilinde sık kullanılan temel operatörler incelenecektir.

OPERATÖR NEDİR?

Değişkenler üzerinde işlem yapabilmek için operatör kullanılır.

Operatörler, kendi başlarına bir anlam ifade etmeyen ancak başta matematiksel ve mantıksal olmak üzere çeşitli işlemleri gerçekleştirmek için kullanılan özel karakter ya da sembollerdir. Örneğin "a+b" ifadesinde "+" işareti bir operatördür. Operatörler işlevlerini yerine getirebilmek için değişkenlere ihtiyaç duymaktadır. Operatörlerin etki ettiği değişkenlere "operand" denir. Örneğin "a+b" ifadesinde "+" operatörünün etki ettiği operandlar "a" ve "b" dir. Programlama dillerinde kullanılan operatörler işlevlerine göre farklı sınıflara ayrılmaktadır. Aşağıda verilen tabloda (Tablo 7.1) C# içerisinde kullanılan operatörlerin sınıflandırılmış hali görülmektedir.

Tablo 7.1. Operatör Sınıflandırmaları

Operatörler	C# Programlama Dili Sembolleri
Aritmetik Operatörler	+ , - , * , /, ++ , -- , %
Atama Operatörleri	*=, /=, +=, -=, &=, ^=, |=
Mantıksal operatörler	&&, !, ||
Karşılaştırma Operatörü	>, <, >=, <=, ==, !=
Bitsel Operatörler	&, ~, |, ^
Özel Amaçlı Operatörler	TypeOf, Sizeof, New

Tablo 7.1'de yer alan "Özel Amaçlı Operatörler" ileri seviyede bir konu olduğu için bu ünitede anlatılmayacaktır. Ünitede sırasıyla *aritmetik operatörler,*

atama operatörleri, mantıksal operatörler, karşılaştırma operatörleri ve bitsel operatörler incelenecektir.

ARİTMETİK OPERATÖRLER

Toplama, çıkarma, çarpma, bölme ve mod gibi temel matematiksel işlemleri gerçekleştirmek için aritmetik operatörler kullanılır. Aritmetik operatörlerden olan "+" ve "-" işaret operatörü olarak da isimlendirilmektedir. Sayısal değerlerin başına "-" operatörü koyulması ile sabitin veya değişkenin negatif olduğu belirtilir. Pozitif olan sayılar için isteğe bağlı olarak sayının başına"+" işareti konulmaktadır. Önünde işaret operatörü olmayan sayılar varsayılan olarak pozitif kabul edilmektedir. Toplama ve işaret operatörü olarak da kullanılan "+" işaretine en az iki "string" veya "char" "operand" verilmesi durumunda bu iki değer birleşip tek bir değer haline gelmektedir. Ayrıca sayısal değerleri birer birer artırmak ya da azaltmak amacı ile "++" ve "--" operatörleri de kullanılmaktadır.

Aritmetik operatörler kendi içerisinde işlem önceliğine sahiptir. Matematiksel kurallar bu operatörler üzerinde aynı şekilde uygulanmaktadır. Örneğin çarpma işlemi toplama işleminden önceliklidir. Eğer iç içe geçmiş parantezler varsa önceliğe sahip ilk işlem en içteki parantezdir.

Aşağıda her bir aritmetik operatörün kullanımına ait birer örnek verilmiştir:

Toplama Operatörü: (+)

"+" operatörü sayesinde sabit veya değişkenlerin toplama işlemleri gerçekleştirilir. Aşağıda verilen örnekte üç farklı sayının toplama işlemi operatör ile yapılmış ve sonuç çıktısı alınmıştır. Çıktının "messagebox.show" ile gösterilebilmesi için verinin "string" değere dönüştürülmesi zorunludur.

Örnek

```
•static void Main()
• {
•   int a = 10;
•   int b = 11;
•   int c = 12;
•   int topla = a + b + c;
•   MessageBox.Show(topla.ToString());  //ekran çıktısı=> 33
• }
```

Çıkarma Operatörü: (-)

"-" operatörü sayesinde sabit veya değişkenlerin çıkarma işlemleri gerçekleştirilir. Aşağıda verilen örnekte iki farklı sayının çıkarma işlemi yapılmış ve sonuç çıktısı alınmıştır. Örnekte bulunan "b" değeri operatör ile bağdaştırılmadığı için herhangi bir işlev kazanmamıştır.

Örnek

```
•static void Main()
• {
•   int a = 10;
•   int b = 11;
•   int c = 12;
•   int cikarma= c - a;
•   MessageBox.Show(cikarma.ToString()); //ekran çıktısı=>  2
• }
```

Çarpma Operatörü: (*)

"*" operatörü sayesinde sabit veya değişkenlerin çarpma işlemleri gerçekleştirilir. Aşağıda verilen örnekte iki farklı sayının çarpma işlemi yapılmış ve sonuç çıktısı alınmıştır. Örnekte bulunan "c" değeri operatör ile bağdaştırılmadığı için herhangi bir işlev kazanmamıştır.

Örnek

```
•static void Main()
• {
•   int a = 10;
•   int b = 11;
•   int c = 12;
•   int carpma = a * b;
•   MessageBox.Show(carpma.ToString()); //ekran
 çıktısı=>110
• }
```

Bölme Operatörü: (/)

"/"operatörü sayesinde sabit veya değişkenlerin bölme işlemleri gerçekleştirilir. Aşağıda verilen örnekte "c" sayısının ikiye bölme işlemi yapılmış ve sonuç çıktısı alınmıştır. Örnekte bulunan "a" ve "b" değeri operatör ile bağdaştırılmadığı için herhangi bir işlev kazanmamıştır.

Örnek

```
•static void Main()
• {
•   int a = 10;
•   int b = 11;
•   int c = 12;
•   int bolme = c / 2;
•   MessageBox.Show(bolme.ToString());  //ekran çıktısı=>  6
• }
```

Mod Operatörü: (%)

"%"operatörü sabit veya değişkenlerin mod işlemleri gerçekleştirilir. Mod işlemi matematiksel bir işlemdir. Bir sayının bir sayıya bölümünden kalan sayıya mod alma denir. Aşağıda verilen örnekte "c" sayısının "a" sayısına göre mod işlemi yapılmıştır.

Örnek

```
static void Main()
{
  int a = 10;
  int b = 11;
  int c = 12;
  int mod = c % a;
  MessageBox.Show(mod.ToString());    //ekran çıktısı=>  2
}
```

İşaret Operatörü:

Bu operatör sabit veya değişkene atanan değerin matematiksel olarak yön değiştirmesini sağlar. Aşağıda verine örnekte "a" değerini negatif yönde değiştirme işlemi gerçekleştirilmiştir.

Örnek

```
static void Main()
{
  int a = 10;
  MessageBox.Show((-a).ToString());    // ekran çıktısı => -10
  MessageBox.Show((-(-a)).ToString()); // ekran çıktısı =>  10
}
```

String Birleştirme Operatörü:

Birleştirme operatörü sayesinde "string" veya "char" değere sahip veriler bir araya getirilebilir. Birleştirme operatörünün çalışabilmesi için en az iki değerin birleştirilmesi gerekmektedir. Aşağıda verilen örnekte birbirinden farklı üç veri tek veri haline getirilmiştir.

Örnek

```
static void Main()
{
  MessageBox.Show("Merhaba " + "Yazılım " +
 "Öğreniyorum");
}
```

Artırma Azaltma Operatörleri: (++, --)

Artırma/azaltma operatörleri "integer" değerler üzerinde uygulanabilir. Matematiksel olarak değişkene ait değerin bir azaltıp ya da bir artırılması için kullanılan operatörlerdir. Aşağıda örnekte verilen sayi değeri "++" operatörü sayesinde bir değer artmıştır, "--" operatörü ile de bir değer azalmıştır.

Örnek

```
•static void Main()
• {
•   int sayi = 10;
•   sayi++;
•   MessageBox.Show(sayi.ToString()); //ekran çıktısı => 11
•   sayi--;
•   MessageBox.Show(sayi.ToString()); //ekran çıktısı => 9
• }
```

ATAMA OPERATÖRLERİ

Atama operatörü, bir değişkene değer atamak amacıyla kullanılır. Atama operatörlerinin temel karakteri "="'dir. Aşağıdaki örnekte görüldüğü üzere "=" operatörü kullanılarak 6 değeri "sayi" değişkenine atanmıştır.

Örnek

```
•static void Main()
• {
•   int sayi = 6;
• }
```

Bazı durumlarda değişkenin var olan değeri üzerinde toplama, çıkarma gibi işlemler yapılabilir. Bu gibi durumlarda eşittir *(=)* operatörünün soluna istenilen aritmetik işlemi temsil eden operatör konur. Bu tip *operatörlere işlemli atama operatörü* de denebilir. Aşağıda verilen örnekte işlemli atama operatörünün nasıl çalıştığına yönelik bir örnek görülmektedir:

Örnek

```
•static void Main()
• {
•   int a = 10;
•   int b = 11;
•   a+ = b
• }
```

Yukarıda “a” ve “b” değişkenleri tanımlanmış ardından “a” isimli değişkene “b” isimli değişkenin değeri *“+=”* operatörü kullanılarak eklenmiştir. Bu işlem sonucunda “a” değişkeni 21 değerini alır.

İşlemli atama operatörleri kullanım şekline bağlı olarak tasarlanan kodun daha kısa ve sade olmasını sağlar. Aşağıdaki tabloda (Tablo 7.2) farklı operatörlerin kullanıldığı aynı çıktıyı üreten kod örnekleri verilmiştir.

Tablo 7.2. Atama Operatörü ile ilgili Örnekler

İŞLEM	İŞLEMLİ ATAMA ÖRNEĞİ
A = A + B	A+ = B
A = A - B	A- = B
A = A * B	A* = B
A = A / B	A/ = B

MANTIKSAL OPERATÖRLER

Sıklıkla kullanılan operatör gruplarından biri de mantıksal operatörlerdir. Özellikle program akışını yönlendirirken kurulan döngü ve kontrol yapılarında mantıksal operatörlerden yararlanılır.

Elde edilmek istenilen verinin mantıksal olarak sınanması bu operatörler sayesinde gerçekleştirilir. Mantıksal sınanma, karar verme mekanizması ile doğru orantılıdır. Karar verme mekanizmasında karşımıza iki farklı durum çıkmaktadır. Örneğin a sabiti b sabitinden büyük ise x işlemi değil ise y işlemini gerçekleştir. Bu işlemlerin gerçekleşmesi için gerekli döngüler kurulurken döngü içerisinde mantıksal operatörlerden yararlanılması gerekmektedir. Mantıksal operatörler sadece “bool” tipindeki değerler ile işlem yapabilir. Bu operatörlerle yapılan karşılaştırmaların sonuçları “True” (Doğru) veya “False” (Yanlış) olması durumuna göre program akışı belirlenir. Başlıca mantıksal operatörler aşağıdaki gibidir:

&& (ve) Operatörü:

“&&” operatörü sayesinde birden fazla koşulun sınanması gerçekleştirilir. Operatörün işlem yapabilmesi için sınanan yani karşılaştırılan her koşulun doğru (true) olması gerekmektedir. Tablo 7.3.’de “&&” operatörünün çalışma mantığı gösterilmiştir.

Aşağıdaki örnekte “&&” operatörünün çalışma mantığını gösteren örnek verilmiştir.

Tablo 7.3. && Operatörü

İfade-1	İfade-2	İfade-1 && İfade-2
False	False	False
False	True	False
True	False	False
True	True	True

Örnek

```
•static void Main()
• {
•   bool a = true && true;
•   bool b = true && false;
•   bool c = false && true;
•   bool d = false && false;
•   MessageBox.Show(a.ToString()); // Ekran çıktısı true
•   MessageBox.Show(b.ToString()); // Ekran çıktısı false
•   MessageBox.Show(c.ToString()); // Ekran çıktısı false
•   MessageBox.Show(d.ToString()); // Ekran çıktısı false
• }
```

|| (veya) Operatörü:

Operatörlerin çalışma mantığı birbirlerine benzemektedir. '|| operatörü' de mantıksal sınama yani karar verme aşamasında kullanılır. Karar verme aşamasında iki farklı durumdan en az birinin true (doğru) olması, kriteri sağlama amacı taşır. Tablo 7.4.'de "||" operatörünün çalışma mantığı gösterilmiştir.

Tablo 7.4. || Operatörü

İfade 1	İfade 2	İfade-1 || İfade-2
False	False	False
False	True	True
True	False	True
True	True	True

Örnek

```
•static void Main()
• {
•   bool a = true || true;
•   bool b = true || false;
•   bool c = false || true;
•   bool d = false || false;
•   MessageBox.Show(a.ToString()); // Ekran çıktısı true
•   MessageBox.Show(b.ToString()); // Ekran çıktısı true
•   MessageBox.Show(c.ToString()); // Ekran çıktısı true
•   MessageBox.Show(d.ToString());// Ekran çıktısı false
• }
```

! Operatörü:

"!" Operatörü sayesinde "true" olan bir değer "false", "false" olan bir değer "true" olarak değiştirilir. Değişkenin değeri tam tersi değer ile değiştirilerek kontrol edilir.

Örnek

```
•static void Main()
• {
•   bool a = !true;
•   bool b = !false;
•   bool c = (!(4<3));
•   MessageBox.Show(a.ToString()); // Ekran çıktısı False
•   MessageBox.Show(b.ToString()); // Ekran çıktısı True
•   MessageBox.Show(c.ToString()); // Ekran çıktısı True
• }
```

KARŞILAŞTIRMA OPERATÖRLERİ

Karşılaştırma operatörleri genellikle koşul ifadelerinde kullanılır. Koşul ifadelerinde bulunan verilerin değerleri arasında karşılaştırma yaparak değerler arasında büyüklük, küçüklük ve eşitlik gibi durumlar incelenir. Karşılaştırma operatörlerinin öncelikleri aritmetik operatörlerden daha düşüktür. Ayrıca karşılaştırma operatörleri kendi içerisinde de öncelik durumlarına göre ayrılmaktadır. Bunlar arasında "<", ">", "<=", ">=" operatörleri, "==" ve "!=" operatörlerine göre önceliklidir. Karşılaştırma operatörlerinden olan "==" ve "!=" haricindeki diğer operatörler, "string" değerler üzerinde uygulanamaz. Operatörleri ve anlamları tablo 7.5 de verilmiştir:

Tablo 7.5. Karşılaştırma Operatörleri

Operatör		Anlamı	Örnek (Mantıksal)	Örnek (Matematiksel)
==	→	Eşit	Deger=(7==5) Sonuc=False	A==B*10
>	→	Büyük	Deger=(5>3) Sonuc=True	(A^2)>3
>=	→	Büyük Eşit	Deger=(3>=5) Sonuc=False	A*B>=C
<	→	Küçük	Deger=(3<5) Sonuc=True	A<B
<=	→	Küçük Eşit	Deger=(3<=3) Sonuc=True	35<=A
!=	→	Eşit Değil	Deger=(3!=5) Sonuc=True	AB!=(C*56)

Aşağıda karşılaştırma operatörünün kullanıma ait örnek verilmiştir. Örnek içinde kullanıcıdan istenen bir sayının 5 den büyük, küçük ve eşit olması durumu incelenmiştir.

Örnek

```
•static void Main(string[] args)
• {
•   int number = 5;
•   if (number > 5)
• {      MessageBox.Show("Girdiginiz sayi 5'ten Büyüktür.");
• }
•   else if (number < 5)
• {      MessageBox.Show("Girdiginiz sayi 5'ten Kücüktür.");
• }
•  else
• {      MessageBox.Show("Girdiginiz Sayi 5'tir");
• }
• } // Ekran çıktısı => "Girdiğiniz sayi 5'tir"
```

BİTSEL OPERATÖRLER

Bitsel operatörler "&" (bitsel ve), "~" (bitsel değil), "|" (bitsel veya), "^" (bitsel özel veya) operatörleridir. Sayıların kendileri yerine bitlerini yani ikilik tabandaki karşılıklarını kullanan operatörlerdir. Bitsel operatörlerin kullanılabilmesi için değerlerin tam sayı olması gerekmektedir.

"&" (ve) operatörü incelendiğinde çalışma mantığında karşılıklı sayıların 1 olması durumunda ilgi basamak 1 diğer durumlarda 0 olarak saklanmaktadır.

"~" (değil) operatörünün çalışma mantığında ise ikilik sayı sisteminde bulunan her basamağın tam tersi saklanır.

"|" (veya) operatöründe karşılıklı sayıların herhangi birisinin 1 olması durumu aranır ve ilgili basamak 1 olarak saklanır diğer durumlarda 0 olarak saklanır.

"^" (özel veya) operatörü karşılıklı basamakların farklı olması durumunda ilgili basamak 1 diğer durumlarda 0 olarak saklanır.

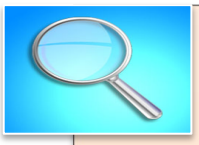
Aşağıda bitsel operatörlerin kullanımına yönelik bir örnek verilmiştir.

Örnek

```
•static void Main()
• {
•   byte a =5 & 3; // 00000101 & 00000011  -->       çıktı 00000001 (1)
•   byte b =5 | 3; //  00000101 | 00000011  -->       çıktı 00000111 (7)
•   byte c = 5 ^ 3; //  00000101 ^ 00000011  -->       çıktı 00000110 (6)
•   byte h = 5;
•   byte e = (byte) ~ d;       //  00000101'in değili   --> çıktı 11111010 (250)
•   byte f = (byte) ~ (a+b); //  00001000'ın değili   --> çıktı 11110111 (247)
•   byte g = (byte) ~ (a+7); // 00001000'ın değili   --> çıktı 11110111 (247)
•   MessageBox.Show(a.ToString() + " " + b.ToString() + " " + c.ToString()
  + " " + h.ToString() + " " + f.ToString() + " " + g.ToString());
• }
```

Bireysel Etkinlik

- Karşılaştırma operatörleri kullanarak A,B ve C sayılarının hangisinin daha büyük olduğunu bulunuz.
- Operatörleri kullanarak A ve B sayısını çarpıp 5 sayısı ile modunu alınız çıkan sonucu artırma operatörü sayesinde 1 artırınız.
- Bitsel operatörlerin çalışma mantığını anlayabilmek için 2 lik tabanda sayı dönüşümlerini öğreniniz.



Özet

- Programlama dillerinde kullanılan temel yapılardan birisi operatörlerdir. Operatörler hesaplamalar, mantıksal işlemler ya da kontrol fonksiyonları gibi birçok yerde kullanılmaktadır.
- Operatörler, kendi başlarına bir anlam ifade etmeyen ancak başta matematiksel ve mantıksal olmak üzere çeşitli işlemleri gerçekleştirmek için kullanılan özel karakter ya da sembollerdir.
- Programlama dillerinde kullanılan operatörler işlevlerine göre farklı sınıflara ayrılmaktadır. Operatörler incelendiğinde bu sınıflandırma aritmetik, atama, mantıksal, karşılaştırma, bitsel ve özel amaçlı operatörler olarak karşımıza çıkmaktadır.
  - *Toplama, çıkarma, çarpma, bölme ve mod gibi temel matematiksel işlemleri gerçekleştirmek için aritmetik operatörler kullanılır.*
  - Atama operatörleri bir değişkene değer atamak amacı ile kullanılır değer atamanı temel operatörü "=" ' dir.
  - Sıklıkla kullanılan operatör gruplarından biri de mantıksal operatörlerdir. Özellikle program akışını yönlendirirken kurulan döngü ve kontrol yapılarında mantıksal operatörlerden yararlanılır.
  - Karşılaştırma operatörleri genellikle koşul ifadelerinde kullanılır. Koşul ifadelerinde bulunan verilerin değerleri arasında karşılaştırma yaparak değerler arasında büyüklük, küçüklük ve eşitlik gibi durumları inceler.
  - Bitsel operatörler "&" (bitsel ve), "~" (bitsel değil), "|" (bitsel veya), "^" (bitsel özel veya) operatörleridir. Sayıların kendileri yerine bitlerini yani ikilik tabanda ki karşılıklarını kullanan operatörlerdir.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi aritmetik operatör değildir?
   a) +
   b) Mod
   c) /
   d) *
   e) -

2. Bir değişkene değer atamak için aşağıdaki operatörlerden hangisi kullanılır?
   a) +
   b) *
   c) -
   d) =
   e) !

3. Aşağıdakilerden hangisi c#' da kullanılan operatörlerden biri değildir?
   a) And
   b) &&
   c) +=
   d) *
   e) -

4. Aşağıdakilerden hangisi işlemli atama operatörü değildir?
   a) <>
   b) +=
   c) -=
   d) *=
   e) /=

5. C#' da eşit değildir anlamına gelen operatör aşağıdakilerden hangisidir?
   a) >
   b) !=
   c) <>
   d) ?=
   e) +=

Toplama, çıkarma, çarpma, bölme, mod ve üs alma gibi temel matematiksel işlemleri gerçekleştirmek için ..... operatörleri kullanılır.

6. Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?
   a) Karşılaştırma
   b) Mantıksal
   c) Atama
   d) Aritmetik
   e) İşlemli Atama

7. String tipindeki karakterleri birleştirmek için aşağıdaki operatörlerden hangisi kullanılabilir?
   a) ‘’
   b) *
   c) -=
   d) %
   e) +

8. Mantıksal operatörler hangi tipte değer ile kullanılabilir?
   a) Boolean
   b) String
   c) Float
   d) Double
   e) Integer

9. Değişkenler veya ifadeler arasındaki büyüklük, küçüklük gibi derecelerin belirlenmesi için hangi operatörler kullanılır?
   a) Mantıksal
   b) Aritmetik
   c) Karşılaştırma
   d) Özel Amaçlı
   e) Atama

10. Aşağıdakilerden hangisi c#’ da kullanılan mantıksal operatördür?
    a) Mod
    b) Or
    c) And
    d) Not
    e) !

Cevap Anahtarı

1.b, 2.d, 3.a, 4.a, 5.b, 6.d, 7.e, 8.a, 9.c, 10.e

YARARLANILAN KAYNAKLAR

ALGAN, Sefer, (2009), Her Yönüyle C#, Pusula Yayıncılık, İstanbul.

AKTAŞ, Volkan, (2013), Her Yönüyle C# 5.0, KODLAB, İstanbul.

KIZILÖREN, Tevfik, (2013), Her Yönüyle C, KODLAB, İstanbul.Türk Medeni Kanunu, (2001). 12 Nisan 2018 tarihinde