C
Operatörler

Dr. Öğr. Üyesi M. Ozan AKI

Rev 1.0

Operatörler

Bir veya iki değişken ya da sabit arasında işlem yaparak yeni bir değer üreten ya da mevcut bir değişkenin değerinin değiştirebilen, işlem yapan sembollerdir.

- Atama Operatörü
- Aritmetik Operatörler
- İlişkisel Operatörler
- Mantıksal Operatörler
- Bitsel işlem yapan Operatörler
- Özel amaçlı Operatörler

Atama Operatörü (=)

Sağındaki deyim ya da sabitin değerini, solundaki değişkene atayan opetaördür.

<değişken adı> = <deyim>;

basit atamalar:

int a = 123;

double d = 1.23;

char c = ‘A’;

Atama Operatörü (=)

Eğer atama operatörünün sağındaki değişken ya da sabitin tipi, solundaki hedef tipten farklı ise burada otomatik olarak tip dönüşümü yapılır.

Eğer, sağ tarafın bit sayısı sol tarafınkinden fazla ise, sol üst bitler kesilir ve alt sağ bitler sol taraftaki değişkene atanır.

Her bir atama ifadesi, bir bütün olarak değer döndürür.

Aritmetik Operatörler (**+ - * / % -- ++**)

+, -, * , / operatörleri, bildiğimiz dört işlem yapar. İşleme giren veri tipinde bir değer döndürür.

/ operatörü tam sayılar üzerinde kullanıldığında, eğer bölme işleminde kalan varsa atılır, sadece tam kısmı döndürülür. (sayı yuvarlanmaz!!!)

```
a = 5 / 2;
b = 10 / 3;
```

Aritmetik Operatörler (**+ - * / % -- ++**)

% Modül operatörü, tam sayı bölme işleminde kalanı verir. float ve double tipinde kullanılmaz.

a = 5 % 2;

b = 10 % 2;

c= 10 % 4;

Aritmetik Operatörler (**+ - * / % -- ++**)

+ + ve - - operatörleri, arttırma ve eksiltme operatörleridir. Tek bir değişken üzerine uygulanır, bu değişkenin değerini değiştirir.

a++; işlemi, a = a + 1; işlemi ile aynı sonucu verir.

b--; işlemi, b = b – 1; işlemi ile aynı sonucu verir.

Aritmetik Operatörler (**+ - * / % -- ++**)

+ + ve - - operatörleri, değişkenin hem sağına hem soluna yazılabilir.

a++; ile ++a; ifadeleri tek başına kullanıldığında hiç bir fark yoktur.

Ancak bir ifade içerisinde kullanıldığında;

operatör değişkenin solunda ise, önce değişkenin değeri bir arttırılır/eksiltir, daha sonra ifade içerisndeki işleme sokulur.

operatör değişkenin sağında ise, önce değişkenin değeri ifade içerisinde işleme sokulur, işlem bittikten sonra değeri bir arttırılır/eksiltilir.

Aritmetik Operatörler (**+ - * / % -- ++**)

```
int a,b;

a = 5;
b = ++a;
printf("a=%d b=%d\n",a,b);

a = 5;
b = a++;
printf("a=%d b=%d\n",a,b);
```

Aritmetik Operatörler (+ - * / % -- ++)

Aritmetik Operatörlerde işlem önceliği;

En Yüksek Öncelik

++ --

- (tek eksi, işaret)

* / %

+ -

En Düşük Öncelik

```
int a,b,c,d;
a= b+c++/(-c*++a%3);
```

İlişkisel ve Mantıksal Operatörler (<, <=, >, >=, ==, !=, &&, ||, !)

C Dilinde **Doğru** (**True**) ve **Yanlış** (**False**)

C#, Java, C++ gibi üst düzey diller hatta C99 ile belirlenen standartlarda mantıksal işlemler için bool (Boolean) tipinde bir veri tipi tanımlanmıştır.

Bu veri tipi sadece **true** ve **false** değerlerini alabilir.

İlişkisel operatörler, bu mantıksal sonuçlardan birini döndür.

Mantıksal Operatörler ile, bu mantıksal sonuçlar üzerinde mantıksal işlem yaparak yine mantıksal bir sonuç üretirler.

İlişkisel ve Mantıksal Operatörler (<, <=, >, >=, ==, !=, &&, ||, !)

C Dilinde **Doğru** (**True**) ve **Yanlış** (**False**)

Ancak C dilinde özel bir mantıksal veri tipi yoktur.

Bunun yerine, Herhangi tam sayı olabilen değer;

- **sıfır ise FALSE (Yanlış),**
- **sıfırdan farklı ise TRUE (Doğru)**

kabul edilir.

İlişkisel ve Mantıksal Operatörler (<, <=, >, >=, ==, !=, &&, ||, !)

0 -> false

3 -> true

38 < 76 -> true

58 <= 58 -> true

17 < 17 -> false

27 > -89 -> true

43 == 65 -> false

23 != 12 -> true

32 == 32 -> true

22 != 22 -> false

-25 < -12 -> true

0 > 0 -> false

0 == 0 -> true

0 != 0 -> false

İlişkisel ve Mantıksal Operatörler (<, <=, >, >=, ==, !=, &&, ||, !)

Birden fazla mantıksal durumu sınamak ya da mantıksal durumlar arasında işlem yapmak amacıyla mantıksal operatörler kullanılır.

&& : VE (AND) opertörü; sadece her iki mantıksal durum doğru (true) ise doğru (true) döndürür.

|| : VEYA (OR) operatörü; her iki mantıksal durumdan herhangi biri doğru (true) ise doğru (true) döndürür.

! : DEĞİL (NOT) operatörü; Önüne geldiği mantıksal durumu tersine çevirir. Doğru (true) ise Yanlış (false), Yanlış (false) ise doğru (true) yapar.

İlişkisel ve Mantıksal Operatörler (<, <=, >, >=, ==, !=, &&, ||, !)

(1 < 2) && (4 < 7) → true

(5 == 4) || (8 != 4) → true

(32 <= 40) && !(43 > 55) → true

((a < 23) || (b > 47)) && (c != 0)

(a < b) && (b < c)

(a == b) && (b != c) → (a == b) && !(b == c)

Bitsel İşlem Yapan Operatörler (~ , & , | , ^ , << , >>)

Bitsel işlem yapan operatörler, operans olarak verilen değişkenlerin bitleri üzerinde işlem yapan operatörlerdir.

Bu operatörlerin çalışmasını kağıt üzerinde çözebilmek için sayıları bitlerine ayırmamız gerekir.

Bitsel İşlem Yapan Operatörler (~ , & , | , ^ , << , >>)

~ : DEĞİL (NOT) operatörü; önüne geldiği değerin tüm bitlerini tersine çevirir, 0'ları 1, 1'leri 0 yapar.

& : VE (AND) operatörü; iki değer arasında bit bit lojik VE işlemi yapar.

| : VEYA (OR) operatörü; iki değer arasında bit bit lojik VEYA işlemi yapar.

^ : ÖZEL-VEYA (XOR) opratörü; iki değer arasında bit bit lojik ÖZEL-VEYA işlemi yapar.

<< : solundaki değerin bitlerini, sağındaki değer kadar **sola** kaydırır.

>> : solundaki değerin bitlerini, sağındaki değer kadar **sağa** kaydırır.

Bitsel İşlem Yapan Operatörler
(~ , & , | , ^ , << , >>)

~ : DEĞİL (NOT) operatörü

unsigned char a,b;

a=50; // 50 → 00110010

b = ~a; // ~50 → 11001101 → 205 *(255-50)*

printf(“b=%d\n”, b);

Bitsel İşlem Yapan Operatörler
(~ , & , | , ^ , << , >>)

& : VE (AND) operatörü;

unsigned char a,b,c;

a=50; b=38; // 50 → 00110010

c = a & b; // 38 → 00100110

// c = 00100010 →34

printf("c=%d\n", b);

Bitsel İşlem Yapan Operatörler
(~ , & , | , ^ , << , >>)

| : VEYA (OR) operatörü;

```
unsigned char a,b,c;
a=50; b=38; // 50 → 00110010
c = a | b;      // 38 → 00100110
                //   c =   00110110 →54
printf("c=%d\n", b);
```

Bitsel İşlem Yapan Operatörler
(~ , & , | , ^ , << , >>)

^ : ÖZEL-VEYA (XOR) opratörü;

```
unsigned char a,b,c;
a=50; b=38; // 50 → 00110010
c = a ^ b;      // 38 → 00100110
                //   c =    00010100 →20
printf("c=%d\n", b);
```

Bitsel İşlem Yapan Operatörler
(~ , & , | , ^ , << , >>)

<< : Operatörü

```
unsigned char a,b,c;
a=6; b=3;      // 6 → 00000110
c = a << b;      // c = 00110000  → 48

printf("c=%d\n", b);
```

// Bir sayının bitlerini bir bit sola kaydırmak,
// o sayıyı 2 ile çarpmak demektir.

Bitsel İşlem Yapan Operatörler
(~ , & , | , ^ , << , >>)

>> : Operatörü

unsigned char a,b,c;
a=50; b=3; // 50 → 00110010
c = a >> b; // c = 00000110 → 6

printf("c=%d\n", b);
// Bir sayının bitlerini bir bit sağa kaydırmak,
// o sayıyı 2'ye bölmek demektir.

Bitsel İşlem Yapan Operatörler ile Bit Test ve Bit Maskeleme

Bazı durumlarda bir değişkenin bitleri bazı bitlerini test etmek ya da bazı bitlerini, diğer bitlere dokunmadan değiştirmek gerekebilir.

Bu durumda bitsel işlem yapan operatörler kullanılarak bit maskeleme yapılır.

Bitsel İşlem Yapan Operatörler ile Bit Test ve Bit Maskeleme

a değişkeninin 3. bitini 1 yapalım

a = a | 0x08; ya da **a = a | (1<<3);**

a değişkeninin 6. bitini 0 yapalım

a = a & ~0x40; ya da **a = a & 0xBF;**

a değişkeninin 4. biti 0 mıdır?

(a & 0x10) == 0; ya da **(a & (1<<4)) == 0;**

a değişkeninin 5. biti 1 mi?

(a & 0x20) == 0x20; ya da **(a & (1<<5)) == (1<<5);**

Özel Amaçlı Operatörler
(? : , sizeof() (cast))

? : Koşullu değerlendirme operatörü

ifade1 **?** ifade2 **:** ifade3

ifade1 doğru (true) ise → ifade2,
yanlış (false) ise → ifade3 geçerli olur.

```
int a,b,min,max,abs,yil,subat;
min = (a < b) ? a : b;
max = (a < b) ? b : a;
abs = (a < 0) ? –a : a;
subat = ( yil % 4 ) ? 28 : 29;
```

Özel Amaçlı Operatörler
(? : , sizeof() (cast))

sizeof() opratörü;

Öntanımlı ya da kullanıcı tanımlı veri tiplerinin bellekte kapladığı alanı **byte** cinsinden döndür.

sizeof(char) → 1
sizeof(int) → 4
sizeof(1.2) → 8 (double)
sizeof(doube) → 8
sizeof('A') → 1
sizeof(main()) → 2 (pointer)

Özel Amaçlı Operatörler
(? : , sizeof() (cast))

(cast) opratörü;

Atama ve işlemlerde sabit sayı ya da ifadenin C tarafından varsayılan veri tipini değiştirmek amacıyla kullanılır.

(Tip Bildirimi) İfade

Özel Amaçlı Operatörler
(? : , sizeof() (cast))

(cast) opratörü;

```
float f;
f = 10 / 3;                 // f → 3
f = (float) 10 / 3;         // f → 3.33
f = (float) (10 / 3);       // f → 3.00
f = 10.0 / 3;               // f → 3.33
```

Bileşik Atama Operatörleri
(+= -= &= |=)

Bazı Aritmetik ve Bitsel işlem yapan operatörler atama (=) operatörü ile birlikte kullanılarak kısa ifadeler elde edilebilir;

a = a + 1; → yerine → a += 1;

a = a – 1; → yerine → a -= 1;

a = a | 8; → yerine → a |= 8;

a = a & 15; → yerine → a &= 15;

Operatörlerde İşlem Önceliği

Sembol	Operasyon
! , ~ , ++ , --	Lojik NOT , Bitsel NOT , Bir Arttırma, Bir Eksitme
- , (tip)	İşaret Değiştirme , (cast)
*** , &**	Pointer (Değer, Adres)
sizeof()	Bellek alanı
*** , /**	Çarpma, Bölme
%	Modüler Bölme
+ , -	Toplama Çıkarma
<< , >>	Bit Kaydırma
< , <= , > , >=	Karşılaştırma
== , !=	Karşılaştırma (Eşitlik, Farklılık)
& , ^ , |	Bitsel AND , Bitsel XOR , Bitsel OR
&& , ||	Mantıksal AND , Mantıksal OR
? :	Koşullu değerlendirme
= , += , -= , |= , &=	Atama ve operatörle atamalar

Operatörlerde İşlem Önceliği

Eğer işlem önceliğinden emin değilseniz,

(parantez **)**

kullanarak işlem önceliğini garanti altına almaktan hiç çekinmeyin ve işi şansa bırakmayın.

Parantezlerde önce en içteki parantez işlem görür ve en dıştakine doğru devam eder.

Eğer ifadede kullanılan operatörlerin hepsi aynı işlem önceliğine sahipse; İşlem sırası,

SOLDAN SAĞA doğrudur.

Operatörlerde Farklı Veri Tipleri

Eğer işleme giren veri tipleri aynı ise, döndürülen değer yine aynı veri tipinde olacaktır.

Ancak, işleme giren veri tipleri, birbirinden faklı ise, bit uzunluğu küçük olan veri tip, bit uzunluğu daha yüksek olan veri tipine dönüştürülür.

Operatörlerde Farklı Veri Tipleri