6.Ders : C Operatörleri

C programlama, aritmetik, koşullu ve bitsel işlemler de dahil olmak üzere görevleri gerçekleştirmek için çeşitli operatörlere sahiptir. Bu derste, çeşitli C operatörlerini ve bunların nasıl kullanılacağı hakkında bilgi edineceksiniz.

İçindekiler

Operatör, bir değer veya değişken üzerinde çalışan bir semboldür. Örneğin: **+** toplama yapacak bir operatördür.

C, çeşitli işlemleri gerçekleştirmek için geniş bir operatör yelpazesine sahiptir.

Aritmetik Operatörler (Arithmetic Operators)

Bir aritmetik operatör, sayısal değerler üzerinde toplama, çıkarma ve çarpma gibi matematiksel işlemleri gerçekleştirir (sabitler ve değişkenler).

Operatörün Operatör Anlamı

+ toplama veya tekli artış
– çıkarma veya tekli eksilme
* çarpma işlemi
/ bölme işlemi
% Bölme sonrası kalan (mod alma işlemi)

Örnek 1: Aritmetik Operatörler

```
// C Aritmetik operatörlerin çalışmasını gösterme programı
#include <stdio.h>
int main()
{
    int a = 9,b = 4, c;
    c = a+b;
    printf("a+b = %d \n",c);

    c = a-b;
    printf("a-b = %d \n",c);
    c = a*b;
    printf("a*b = %d \n",c);
    c=a/b;
    printf("a/b = %d \n",c);
    c=a%b;
    printf("a mod b işleminden kalan = %d \n",c);
    return 0;
}
```

Çıktısı :

a+b = 13
a-b = 5
a*b = 36
a/b = 2
a mod b işleminden kalan =1

+, – ve * operatörleri, beklediğiniz gibi sırasıyla toplama, çıkarma ve çarpma işlemlerini hesaplar.

Normal hesaplamada, 9/4 = 2.25. Ancak, çıktı programda 2'dir.

Çünkü hem a hem de b değişkenlerinin tamsayılarıdır. Dolayısıyla, çıktı aynı zamanda bir tamsayıdır. Derleyici ondalık noktadan sonraki terimi ihmal eder ve 2.25 yerine 2. cevabı gösterir. Bu konuyu ilerleyen derslerde daha detaylı göreceğiz.

% Mod operatörü kalanı hesaplar. A = 9, b = 4 ile bölündüğünde, kalan 1'dir. % Operatörü yalnızca tamsayılarla kullanılabilir.

Artırma ve azaltma operatörleri (Increment and decrement operators)

Bu operatörün iki işleci var ++ ve – işlenenin değerini (sabit veya değişken) 1 ile değiştirmek için.

Artırma ++ değeri 1 ile artırırken, azalma – değeri 1 ile azaltır. Bu iki operatör, tek bir işlemcide çalıştıkları

anlamına gelir.

Örnek :

```
#include <stdio.h>
int main()
{
    int a = 10, b = 100;
    float c = 10.5, d = 100.5;

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

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

    printf("++c = %f \n", ++c);

    printf("--d = %f \n", --d);

    return 0;
}
```

Çıktısı :

```
++a = 11
--b = 99
++c = 11.500000
++d = 99.500000
```

Atama Operatörleri (Assignment Operators)

```
#include <stdio.h>
int main()
```

```
{
    int a = 5, c;

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

    c += a; // c = c+a
    printf("c = %d \n", c);

    c -= a; // c = c-a
    printf("c = %d \n", c);

    c *= a; // c = c*a
    printf("c = %d \n", c);

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

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

    return 0;
}
```

Çıktısı :

```
c = 5
c = 10
c = 5
c = 25
c = 5
c = 0
```

İlişkisel Operatörler (Relational Operators)

İlişkisel Operatörler, iki işlenen arasındaki ilişkiyi denetler. İlişki doğruysa, 1 (**true**) döndürür; ilişki yanlış ise, 0 (**false**) değerini döndürür.

İlişkisel operatörler **karar verme** ve **döngülerde** kullanılır.

Operator	Operatörün Anlamı	Örnek
==	Eşittir (Equal to)	5 == 3, sonuç 0
>	Büyüktür (Greater than)	5 > 3, sonuç 1
<	Küçüktür (Less than)	5 < 3, sonuç 0
!=	Eşit değil (Not equal to)	5 != 3, sonuç 1
>=	Büyük veya eşittir (Greater than or equal to)	5 >= 3, sonuç 1
<=	Küçük veya eşittir (Less than or equal to)	5 <= 3, sonuç 0

```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10;

    printf("%d == %d = %d \n", a, b, a == b); // true
    printf("%d == %d = %d \n", a, c, a == c); // false

    printf("%d > %d = %d \n", a, b, a > b); //false
    printf("%d > %d = %d \n", a, c, a > c); //false


    printf("%d < %d = %d \n", a, b, a < b); //false
    printf("%d < %d = %d \n", a, c, a < c); //true


    printf("%d != %d = %d \n", a, b, a != b); //false
    printf("%d != %d = %d \n", a, c, a != c); //true
```

```
    printf("%d >= %d = %d \n", a, b, a >= b); //true
    printf("%d >= %d = %d \n", a, c, a >= c); //false


    printf("%d <= %d = %d \n", a, b, a <= b); //true
    printf("%d <= %d = %d \n", a, c, a <= c); //true

    return 0;

}
```

Çıktısı :

```
5 == 5 = 1
5 == 10 = 0
5 > 5 = 0
5 > 10 = 0
5 < 5 = 0
5 < 10 = 1
5 != 5 = 0
5 != 10 = 1
5 >= 5 = 1
5 >= 10 = 0
5 <= 5 = 1
5 <= 10 = 1
```

Mantıksal operatörler (Logical Operators)

Mantıksal operatör içeren bir ifade, ifadenin **doğru** mu **yanlış** mı olduğuna bağlı olarak 0 veya 1 döndürür. Mantıksal operatörler, C programlamasında **karar** vermede yaygın olarak kullanılır.

	Anlamı	Örnek
&&	Mantıksal VE (AND). Yalnızca tüm operandlar doğruysa doğru (true).	Eğer c = 5 ve d = 2 ise, ifade ((c == 5) && (d> 5)) 0'a eşittir.
||	Mantıksal VEYA (OR).Yalnızca bir işlenen doğruysa doğru (true)	Eğer c = 5 ve d = 2 ise, ifade ((c == 5) || (d> 5)) 1'e eşittir.
!	Mantıksal DEĞİL(NOT). İşlenen 0 ise doğru (true) .	Eğer c = 5 ise, ifade! (c == 5), 0'a eşittir.

```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) equals to %d \n", result);

    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) equals to %d \n", result);

    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) equals to %d \n", result);

    result = (a != b) || (c < b);
    printf("(a != b) || (c < b) equals to %d \n", result);

    result = !(a != b);
    printf("!(a == b) equals to %d \n", result);

    result = !(a == b);
    printf("!(a == b) equals to %d \n", result);

    return 0;
}
```

Çıktısı :

```
(a == b) && (c > b) equals to 1
(a == b) && (c < b) equals to 0
(a == b) || (c < b) equals to 1
(a != b) || (c < b) equals to 0
!(a != b) equals to 1
!(a == b) equals to 0
```

Mantıksal operatör programının açıklaması

(a == b) && (c> 5) 1 olarak değerlendirir, çünkü her iki işlenen (a == b) ve (c> b) 1'dir (doğru).
(a == b) && (c <b) 0 olarak değerlendirir, çünkü işlenen (c <b) 0 (yanlış) olur.
(a == b) || (c <b) 1 olarak değerlendirir, çünkü (a = b) 1 (doğru).
(a! = b) || (c <b) 0 olarak değerlendirir, çünkü hem operand (a! = b) ve (c <b) 0'dır (yanlış).
! (a! = b), 1 olarak değerlendirilir, çünkü operand (a! = b), 0 (yanlış) olur. Dolayısıyla,! (A! = B) 1 (gerçek) olur.
! (a == b) 0 olarak değerlendirir, çünkü (a == b) 1 (doğru). Dolayısıyla,! (A == b), 0 (yanlış) olur.

Bitsel Operatörler (Bitwise Operators)

Hesaplama sırasında, toplama, çıkarma, toplama ve bölme gibi matematiksel işlemler, işlemeyi daha hızlı hale getiren ve güç tasarrufu sağlayan **bit düzeyine** dönüştürülür.

Bitsel operatörler, C programlamada **bit seviyesi** işlemleri gerçekleştirmek için kullanılır.

Operatörler	Anlamı
&	Bitsel ve (Bitwise AND)
|	Bitsel veya (Bitwise OR)
^	bitsel özel veya (Bitwise exclusive OR)
~	Bitsel tamamlayıcı (Bitwise complement)
<<	Sola kay (Shift left)
>>	Sağa kay (Shift right)

Diğer Operatörler (Other Operators)

Virgül Operatörü (Comma Operator)

Virgül operatörleri, ilgili ifadeleri birbirine bağlamak için kullanılır. Örneğin:

```
int a, c = 5, d;
```

sizeOf Operatörü (sizeof Operator)

Sizeof, veri boyutunu döndüren tek bir operatördür (sabit, değişkenler, dizi, yapı vb.). Örneğin :

```
#include <stdio.h>
int main()
{
    int a, e[10];
    float b;
    double c;
    char d;
```

```
    printf("Size of int=%lu bytes\n",sizeof(a));
    printf("Size of float=%lu bytes\n",sizeof(b));
    printf("Size of double=%lu bytes\n",sizeof(c));
    printf("Size of char=%lu byte\n",sizeof(d));
    printf("10 öğeye sahip tam sayı tür dizisinin boyutu = %lu
bytes\n", sizeof(e));
    return 0;
}
```

Çıktısı :

```
Size of int = 4 bytes
Size of float = 4 bytes
Size of double = 8 bytes
Size of char = 1 byte
10 öğeye sahip tam sayı tür dizisinin boyutu = 40 bytes
```

Üçlü operatör (Ternary Operator)

Üçlü OPERATÖR , 3 işleç üzerinde çalışan koşullu bir işleçtir.

Koşullu Operatör Sözdizimi (Conditional Operator Syntax)

```
conditionalExpression ? expression1 : expression2
```

Koşullu operatör aşağıdaki gibi çalışır:

-İlk ifade : koşullu İfade ilk önce değerlendirilir. Bu ifade, doğru olup olmadığını 1 olarak ve yanlış ise 0 olarak değerlendirir.
-Koşullu İfade doğruysa, ifade1 değerlendirilir.

-Koşullu İfade yanlışsa, ifade2 değerlendirilir.

Koşullu operatör Örneği :

```
#include <stdio.h>
int main(){
   char Subat;
   int gunler;
   printf("Bu yıl artık yıl ise, 1 girin.: ");
   scanf("%c",&Subat);

   // If test condition (Subat== 'l') doğru ise, günler 29'a
eşit.
   // If test condition (Subat=='l') yanlış ise, 28 güne eşit.
   gunler= (Subat== '1') ? 29 : 28;

   printf("Subat ayı gün sayısı= %d",gunler);
   return 0;
}
```