BİLGİSAYAR PROGRAMLAMA 1

Ders Notu 4– Operatörler

Konya Teknik Üniversitesi
Elektrik – Elektronik Mühendisliği Bölümü

15.03.2024
Konya

Operatörler

Tüm programlama dillerinde bulunan ve kullanılan değişkenlerin değerleri üzerinde matematiksel, mantıksal ya da karşılaştırmaya dayalı işlem yapabilmek için kullanılan dil elemanlarıdır.

C dilinde kullanılan operatörler aşağıdaki gibi sınıflandırılabilir:

- ✓ Atama operatörü
- ✓ Aritmetik operatörler
- ✓ Karşılaştırma operatörleri/ilişkisel operatörler
- ✓ Mantıksal operatörler
- ✓ Bit-tabanlı operatörler
- ✓ İşaretçi operatörleri
- ✓ « sizeof » operatörü

Atama Operatörü (=)

Değişkene değer atamada kullanılan operatördür.

Operatörün sol tarafına değer atanacak değişken, sağ tarafına da bu değişkene atanacak değer yazılmalıdır.

```
int tam_sayi = 32;

int tam_sayi1 = tam_sayi;
```

Aritmetik Operatörler

Operatör	Görevi
+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
%	Mod Alma
++	1 arttırma

Operatör	Görevi
--	1 eksiltme
+=	Toplama Ataması
-=	Çıkarma Ataması
*=	Çarpma Ataması
/=	Bölme Ataması
%=	Mod Ataması

Mod Operatörü (%)

Bu operatör, solunda bulunan değişkenin/değerin sağında bulunan değişkene/değere bölümünden kalanı hesaplar.

```
Bolunen = 45;
Bolen1 = 15;
Bolen2 = 8;
Kalan1=Bolunen%Bolen1;
Kalan2=Bolunen%Bolen2;
```

Arttırma ve Eksiltme Operatörleri (++, --)

Bu operatörler, uygulandığı değişkenin değerini bir arttırır ya da azaltır.

```
a++;     /* a=a+1 */
a--;     /* a=a-1 */
++a;     /* a=a+1 */
--a;     /* a=a-1 */
```

Bir değişkenin sağında ya da solunda bulunabilir. Eğer önce operatör, sonra değişken yazılmışsa; değişkenin değeri bir artırıldıktan sonra atama işlemi yapılır. Tersi durumda ise atama yapıldıktan sonra değişkenin değeri artırılır ve bu durum atama yapılan değişkeni etkilemez.

```
b=a++;     /* b=a; a=a+1; */
d=++a;     /* a=a+1; d=a; */
```

ÖRNEK:

Aşağıdaki kodun ekran çıktısı nasıl olur? Kağıt üzerinde deneyip elde ettiğiniz sonucu not ediniz, bilgisayarda çalıştırdıktan sonra bulduğunuz sonuçla karşılaştırınız.

```
#include<stdio.h>
int a,b,c,d;
int main()
{ a=5;
b=a++;
c=a;
d=++a;
printf("b=%d dir, c=%d dir, d=%d dir, a=%d dir\n",b,c,d,a);
printf("Simdi ise b=%d dir",--b);
return 0;
}
```

Aritmetik Atama Operatörleri (+=, -=, *=, /=, %=)

Aritmetik işlem ve atamanın aynı anda yapılmasını sağlar.

Topla/Çıkar ve Ata: Solundaki değişkenin değerini, sağındaki değişkenin değeri kadar artırır/azaltır.

Çarp/Böl ve Ata: Solundaki değişkenin değerini sağındaki değişkenin değeri ile çarpar ya da değerine böler ve sonucu solundaki değişkene atar.

Mod Ata: Solundaki değişken değerini sağdaki ifadeye böler ve kalanı solundaki değişkene atar.

```
b-=a;     /* b=b-a; */          d*=a;    /* d=d*a; */

b%=a;     /* b=b%a; */
```

Aşağıdaki kodun ekran çıktısı nasıl olur? Kağıt üzerinde deneyip elde ettiğiniz sonucu not ediniz, bilgisayarda çalıştırdıktan sonra bulduğunuz sonuçla karşılaştırınız.

```c
#include <stdio.h>

main()
{ int a = 45, b = 2;
a +=b ; b+= 3;
printf("a=%d , b=%d\n",a,b);

a -= b; b -= 3;
printf("a=%d , b=%d\n",a,b);

a *= b; b *= 3;
printf("a=%d , b=%d\n",a,b);

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

return 0;
}
```

Karşılaştırma (İlişkisel) Operatörleri (<, >, <=, >=, ==, !=)

Bu operatörler, sağındaki değişken veya ifade ile solundakini karşılaştırır, kullanılan operatör çeşidine göre **doğru (true, 1)** ya da **yanlış (false, 0)** değerini döndürür.

Ancak C dilinde doğrudan true ve false değerlerini tutacak bir veri tipi tanımlanmadığından sonuçlar 0 ve 1 sayılarıyla ifade edilir. «0'dan farklıysa doğru kabul et» yaklaşımı, çok yaygın bir şekilde kullanılmaktadır.

Aşağıdaki kodun ekran çıktısı nasıl olur? Kağıt üzerinde deneyip elde ettiğiniz sonucu not ediniz, bilgisayarda çalıştırdıktan sonra bulduğunuz sonuçla karşılaştırınız.

```c
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10;
    printf("%d == %d -> %d \n", a, b, a == b);
    printf("%d == %d -> %d \n", a, c, a == c);
    printf("%d > %d -> %d \n", a, b, a > b);
    printf("%d > %d -> %d \n", a, c, a > c);
    printf("%d < %d -> %d \n", a, b, a < b);
    printf("%d < %d -> %d \n", a, c, a < c);
    printf("%d != %d -> %d \n", a, b, a != b);
    printf("%d != %d -> %d \n", a, c, a != c);
    printf("%d >= %d -> %d \n", a, b, a >= b);
    printf("%d >= %d -> %d \n", a, c, a >= c);
    printf("%d <= %d -> %d \n", a, b, a <= b);
    printf("%d <= %d -> %d \n", a, c, a <= c);
    return 0;
}
```

Aritmetik Operatörlerin İşlem Sırası

Açıklama	Operatör	Öncelik
Parantez	(), []	1
Artır*, Azalt*	++, --	2
Çarpma, Bölme, Mod	*, /, %	3
Toplama, Çıkarma	+ , -	4
İlişkisel Operatörler	<, >, <=, >=	5
Eşitlik İnceleme	==, !=	6
Atama	=, +=, -=, *=, /=, %=	7

ÖRNEK:

```
a=10*2+1>=4*4+5==5>5-3*2;
/*a değeri nedir?*/
```

- İ1: 10*2 20
- İ2: 4*4 16
- İ3: 3*2 6
- İ4: İ1+1 21
- İ5: İ2+5 21
- İ6: 5-İ3 -1
- İ7: İ4>=İ5 1
- İ8: 5>İ6 1
- İ9: İ7==İ8 1
- İ10: a=İ9
- Sonuç a=1

Mantıksal Operatörler (!, &&, ||,^)

Genel olarak birden fazla karşılaştırma ifadesinin birleştirilip tek bir koşul ifadesi haline getirilmesinde kullanılır.

Mantıksal Operatörlerin Doğruluk Tablosu							
&&(and)			||(or)			!(not)	
x	y	Sonuç	X	Y	Sonuç	X	sonuç
0	0	**0**	0	0	**0**	0	**1**
0	1	**0**	0	1	**1**	1	**0**
1	0	**0**	1	0	**1**		
1	1	**1**	1	1	**1**		

Mantıksal Operatörler (!, &&, ||,^)

XOR (FARKLI) ^ :

İki değişken birbirinden **farklı**ysa sonuç TRUE

İki değişken birbirinin aynıysa sonuç FALSE,

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

XOR Operatörü ^

```
int a=62;                    // 0011 1110 = 62
int b=15;                    // 0000 1111 = 15
int e=a^b;                   // 0011 0001 = 49
Printf("(a^b)=%u", e);
```

Ekran Çıktısı:
(a^b)=49

NOTLAR

- Mantıksal operatörler, aritmetik operatörlerden sonra işletilir.
- Mantıksal operatörlerin kendi içlerinde işlem sırası DEĞİL, VE, YA DA şeklindedir.
- Yazdığınız karmaşık bir ifadede hangi operatörün daha önce çalıştırılacağından emin değilseniz, önce çalıştırılmasını istediğiniz ifadeyi parantez içine almalısınız.

Aşağıdaki kodun ekran çıktısı nasıl olur? Kağıt üzerinde deneyip elde ettiğiniz sonucu not ediniz, bilgisayarda çalıştırdıktan sonra bulduğunuz sonuçla karşılaştırınız.

```c
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;
    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) -> %d \n", result);
    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) -> %d \n", result);
    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) -> %d \n", result);
    result = (a != b) || (c < b);
    printf("(a != b) || (c < b) -> %d \n", result);
    result = !(a != b);
    printf("!(a != b) -> %d \n", result);
    result = !(a == b);
    printf("!(a == b) -> %d \n", result);
    return 0;
}
```

Complement (Tümleme) Operatörü ~

Uygulandığı değişkenin veya değerinin ikilik tabandaki bitlerini ters çevirir.

0 olan bitleri 1 yapar, 1 olan bitleri 0 yapar.

```c
#include <stdio.h>
#include <stdlib.h>

/* run this program using the console pauser or add your ow

int main() {

    unsigned char ch = 63;          // 0011 1111 = 63
    unsigned char tumleyen = ~ch;   // 1100 0000 = 192
    printf(" %d 'nin tumleyeni: %d ", ch,tumleyen);


    return 0;
}
```

```
 63 'nin tumleyeni: 192
--------------------------------
Process exited after 0.01424 seconds with return value 0
Press any key to continue . . .
```

Bit Tabanlı Sağa/Sola Kaydırma Operatörleri << >>

Bu operatörler solundaki değişken veya değerlerin 2'lik tabandaki karşılıklarını, sağındaki değer kadar sağa veya sola kaydırır.

Sola kaydırma işlemi aslında ilgili değeri 2 katına çıkartır.
Sağa kaydırma işlemi ise ilgili değeri 2'ye böler

Bit Tabanlı Sağa/Sola Kaydırma Operatörleri << >>

```c
#include <stdio.h>
#include <stdlib.h>

/* run this program using the console pauser or add yo

int main() {

    int g=8;        // 0000 1000 = 8
    int saga1kaydir=g>>1;       // 0000 0100 = 4
    int saga2kaydir=g>>2;       // 0000 0010 = 2
    int saga3kaydir=g>>3;       // 0000 0001 = 1
    int saga4kaydir=g>>4;       // 0000 0000 = 0

    printf("(%d >> 1) = %d\n", g, saga1kaydir );
    printf("(%d >> 2) = %d\n", g, saga2kaydir );
    printf("(%d >> 3) = %d\n", g, saga3kaydir );
    printf("(%d >> 4) = %d\n", g, saga4kaydir );

    int sola1kaydir=g<<1;       // 0001 0000 = 16
    int sola2kaydir=g<<2;       // 0010 0000 = 32
    int sola3kaydir=g<<3;       // 0100 0000 = 64
    int sola4kaydir=g<<4;       // 1000 0000 = 128

    printf("(%d << 1) = %d\n", g, sola1kaydir );
    printf("(%d << 2) = %d\n", g, sola2kaydir );
    printf("(%d << 3) = %d\n", g, sola3kaydir );
    printf("(%d << 4) = %d\n", g, sola4kaydir );

    return 0;
}
```

```
(8 >> 1) = 4
(8 >> 2) = 2
(8 >> 3) = 1
(8 >> 4) = 0
(8 << 1) = 16
(8 << 2) = 32
(8 << 3) = 64
(8 << 4) = 128

--------------------------------
Process exited after 0.01396 seconds with return
Press any key to continue . . .
```

? : Operatörü (Şartlı Atama)

- Bir değişkene şartlı atama yapılmasını sağlar.
- C programlama dilinde 3 argüman ile kullanılan tek operatördür.
- Örnek:

ders5.4.c

```c
#include <stdio.h>
#include <stdlib.h>

/* run this program using the console pauser or add your own getch, system("pause") or input loop */

int main() {

    int x=11, y=5, z=0, z1=0, a=7, b=9;

    z = x > y ? a : b; // x degeri y'den buyukse, z'nin degeri a olsun aksi halde b olsun.

    printf("z: %d\n", z);

    z1 = x < y ? a: b; // x degeri y'den kucukse, z'nin degeri a olsun aksi halde b olsun.

    printf("z1: %d\n", z1);

    return 0;
}
```

```
z: 7
z1: 9

--------------------------------
Process exited after 0.0144 seconds with return value 0
Press any key to continue . . .
```

İşaretçi Operatörü * (Pointer)

- * operatörü çarpma işlemini temsil etmektedir. Bunun dışında işaretçi operatörü (pointer) olarak da kullanılmaktadır.
- İlerleyen haftalarda daha kapsamlı olarak Pointer konusunda anlatılacaktır.

Sizeof Operatörü

- Bu operatör bize, bir veri tipinin bellekte kaç bayt yer kapladığını gösterir.
- Bir veri tipinin kapladığı alan negatif olamayacağından bu işlem sonucundaki çıktı her zaman pozitif yani işaretsiz tamsayı döndürür (unsigned int).
- printf fonksiyonu ile ekrana çıktı yazdırılırken %lu biçim niteleyicisi kullanılır.
- Örnek:

```
int ornekSayi = 5;
unsigned int kacBayt = sizeof (ornekSayi);
printf ("Bellekte kapladigi alan: %lu bayttir", kacBayt);
```

ÖDEV:

Aşağıdaki işlemlerin sonuçlarını bulup, C programı yardımıyla sağlamasını yapınız.

a= 80/5-3*2<2+3*2;

b= 15-7*2>=1==19-9*2;

a && b = ?

a || b = ?

!a && b =?