BSM101 Programlama Dilleri I

Hafta 2
Değişken Kavramı ve Temel Operatörler

Doç. Dr. Caner ÖZCAN

Nesne

- Bellekte yer kaplayan ve içeriklerine erişilebilen alanlara **nesne** denir.
- Bir ifadenin nesne olabilmesi için bellekte yer belirtmesi gerekir.
  - a = b + c;
  - d = 100;
  - Yukarıdaki ifadelerde a, b, c, d birer nesnedir.

Nesne

- **Nesnelerin özellikleri**: İsmi, değeri, türü, faaliyet alanı ve ömrü.
- **İsim (name):** Nesneyi temsil eden karakterlerdir.
- **Değer (value):** Nesnelerin tuttuğu bilgidir. İstenildiği zaman değiştirilebilir.
- **Tür (type):** Nesnenin türü, işleme girdiğinde derleyici tarafından nasıl yorumlanacağını belirleyen özelliktir.
  - Programlama dillerinin çoğunda **char** (karakter), **integer** (tamsayı) ve **float** (gerçek sayı) gibi nesne türleri bulunur.

Atama Operatörü

- Bir değeri bir değişkene atar ve C programlamada = ile gösterilir.

- Atama operatörü kullanım şekli:

  nesne = ifade;

- Örnekler:

```
a = 23;
b = a * 10;
toplam = toplam + b;
```

Sol Taraf Değeri (lvalue)

- Nesne belirten ifadeler sol taraf ifadeleridir.
- Bir ifadenin sol taraf ifadesi olması için bellekte yer göstermesi gerekir.
- Örneğin a ve b nesneleri tek başlarına sol taraf nesneleridir.
- a + b ise sol taraf değeri değildir. Sadece a ve b nesnelerinin değerlerinin toplamını gösteren bir sayı belirtir.
- Örneğin, a + b = c yazamayız.

Sağ Taraf Değeri (rvalue)

- Nesne belirtmeyen ifadelerdir. Atama operatörünün sağında yer alırlar.
- Sabitler her zaman sağ taraf değerleridir.
- Örneğin a = 100; ifadesinde a sol taraf 100 ise sağ taraf değeridir.
- 100 = a; şeklinde yazılan ifade yanlıştır.
- Aşağıdaki ifadeler hatalıdır.

```
20 = ...;       /* hata */
c - 4 = ...;    /* hata */
(y) = ...;      /* hata */
m * 2 = ...;    /* hata */
```

Tür Kavramı

- "Veri türü" tanımı "Nesne türü" tanımına göre daha geneldir.
- Veri, bellekte yer göstersin veya göstermesin bütün bilgileri kapsar.
- Hem sabitler hem de nesneler "veri" olarak yorumlanır.
- Bir nesnenin türü denince o nesne içinde tutulan bilginin derleyici tarafından yorumlanış biçimi anlaşılmalıdır.
- Nesne türleri aynı zamanda onların bellekte kapladıkları alan hakkında bilgi verir.

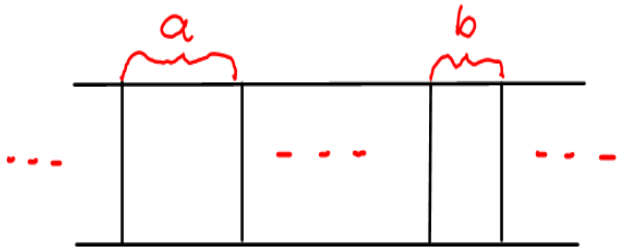
Tür Kavramı

- Hafıza doğrusal bir yapıya sahiptir.
- Nesneler tanımlandıkça hafızada belli bölgelere yerleştirilirler.
- Örneğin "a" ve "b" nesneleri tanımlandığında bellekte bir alana yerleşirler.
- Bellekte kapladıkları alan türe bağlıdır ve farklı olabilir.
- "a" ve "b" birer etikettir ve hafızada bir bölgenin başlangıç adresini ifade eder.

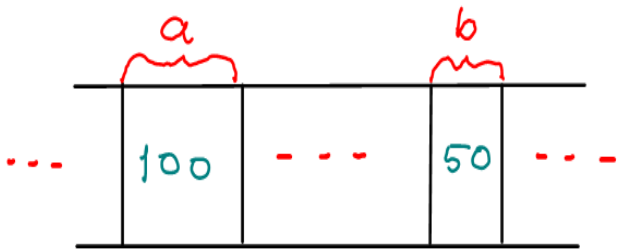
Tür Kavramı

- a = 100 şeklinde bir atama yapıldığında ilgili adres bölgesinde tutulan değer değişir.
- Örneğin a= 100 ve b = 50 şeklinde tanımlanmış iki nesnemiz olsun.
- a = b + 80 ifadesi yalnızca "a" nesnesinin değerini değiştirir ancak "b" nesnesi korunur.

Tür Kavramı



a ve b nesneleri tanımlandı



sol taraf değeri

sağ taraf değeri (sabit)

a = 100 } değer
b = 50 } ataması

a = b + 80

İfade (Expression)

- Bir ifade, bir değeri hesaplayacak matematik bir formüldür ve ";" karakteri ile sonlanır.
  - (a+b)/4;
  - a*b+c
- İfadeler operatörler ile biçimlendirilirler.
- C operatörleri aşağıdakiler gibi sınıflandırılabilir.
  - Atama Operatörü (=)
  - Aritmetik Operatörler (+, -, *, /, %)
  - Aritmetik atama operatörleri (+=, -=, *=, …)
  - Artırma ve Azaltma Operatörleri (++, --)
  - İlişkisel operatörleri (<, <=, ==, >=, >)
  - Mantıksal operatörler (&&, ||, !)

Aritmetik Operatörler

- Aritmetik operatörler binary operatörlerdir (2 tane operand olur).
- 3+7 ifadesinde + binary operatördür ve 3 ile 7 operandlardır.
- * işareti çarpma operatörüdür,
- / işareti bölme operatörüdür. İki tam sayının bölümünün sonucu tam sayıdır.

  Örneğin 7/4 → 1 verir.
- % işareti kalan bulma operatörüdür. Sadece tam sayı operandlar ile kullanılır. Tam bölme sonucu elde edilen kalanı verir.

  Örneğin 7%4 → 3 verir.

Aritmetik Operatörler

İşlem	Aritmetik Operatör
Toplama	+
Çıkarma	-
Çarpma	*
Bölme	/
Kalan	%

Aritmetik Operatörlerde Öncelik Kuralı

SIRA	OPERATOR	İŞLEM
1	( )	Parantez
2	* / %	Çarpma Bölme Kalan
3	+ -	Toplama Çıkarma

Aritmetik Operatörlerde Öncelik Kuralı

- Parantez içindeki ifadeler en yüksek önceliklidir ve ilk önce değerlendirilir.
- İç içe parantez durumunda en içteki parantez içinden başlanır.
  - ((a+b)+c) → Önce a+b toplanır sonra sonuç ile c toplanır.
- Aynı seviyedeki parantezlerde öncelik soldan sağadır.
- Parantezden sonra çarpma, bölme ve kalan bulma operatörleri gelir.
- Son olarak ta toplama, çıkarma gelir.

Aritmetik Operatörlerde Öncelik Kuralı

- Çarpma, bölme ve kalan bulma aynı seviyedeki operatörlerdir.
- Bu operatörler eğer bir ifadede bir arada veya birden fazla kez kullanılıyorlarsa öncelik soldan sağa doğrudur.
- Toplama ile de çıkarma aynı seviyededir. Kendi aralarında öncelik solda sağa doğrudur.
- Öncelik kurallarını hatırlamak zordur.
- Bu yüzden ifadeleri parantez ile ayırmak en iyi yöntemdir.
  - Örnek: sonuc = (a*b) + (a/b);

Aritmetik Operatörlerde Öncelik Kuralı

- Eğer (a+b+c+d+e) toplam değeri 5'e bölünmek isteniyorsa ifade şu şekilde olmalı.
  - m= (a + b + c + d + e) / 5;
- Burada bölme daha öncelikli olduğu için parantez kullanmak zorundayız.
- Parantez kullanılmasaydı m = a+b+c+d+e/5; ifadesinde önce e/5 hesaplanacaktı daha sonra toplamalar yapılacaktı.
- z = p * r % q + w / x – y;

- y = a * x * x + b * x + c;

  a = 2, b = 3, c = 7 ve x = 5

  y = 2 * 5 * 5 + 3 * 5 + 7

  y = 10 * 5 + 3 * 5 + 7

  y = 50 + 3 * 5 + 7

  y = 50 + 15 + 7

  y = 65 + 7

  y = 72

Aritmetik Atama Operatörleri

- Aritmetik atama operatörleri:

  += -= *= /= %= ...

Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
+=	`c += 7`	`c = c + 7`	10 to c
-=	`d -= 4`	`d = d - 4`	1 to d
*=	`e *= 5`	`e = e * 5`	20 to e
/=	`f /= 3`	`f = f / 3`	2 to f
%=	`g %= 9`	`g = g % 9`	3 to g

Unary Artırma ve Azaltma Operatörü

- sonuc = ++a; → önce arttırılır, sonra atanır (preincrement)
- Yandakine denktir:

```
a = a+1;
sonuc = a;
```

- sonuc = --a; → önce eksiltilir, sonra atanır (predecrement)
- Yandakine denktir:

```
a = a-1;
sonuc = a;
```

Unary Artırma ve Azaltma Operatörü

- sonuc = a++; → atanır, sonra arttırılır (postincerement)
- Yandakine denktir:

```
sonuc = a;
a = a+1;
```

- sonuc = a--; → atanır, sonra eksiltilir (postdecrement)
- Yandakine denktir:

```
sonuc = a;
a = a-1;
```

- Değişken herhangi bir ifade içinde değilse önce veya sonra artırma-azaltma arasında bir fark yoktur.

İlişkisel Operatörler

- İlişkisel operatörler iki değeri karşılaştırır ve ilgili operatöre göre doğru veya yanlış True (1) or False (0) olduğuna karar verir.

İlişkisel Operatörü		
==	X == Y	X eşittir Y'ye
!=	X != Y	X eşit değildir Y'ye
>	X > Y	X, Y'den büyüktür
<	X < Y	X, Y'den küçüktür
>=	X >= Y	X, Y'den büyük veya ona eşittir.
<=	X <= Y	X, Y'den küçük veya ona eşittir.

İlişkisel Operatörler

- C de boolean tip yoktur. Bu yüzden tamsayılar kullanılır.
  - Genel kural:
    - "Sıfır → false, diğer sayılar → true"

- Varsayalım a = 1, b = 2, ve c = 3

İfade	Karar	Değer
a < b	True	1
(a + b) >= c	True	1
(b + c) > (a + 5)	False	0
c != 3	False	0
b == 2	True	1

Mantıksal Operatörler

- Operandları üzerinde mantıksal işlem yaparlar.
- Operandlar "True" veya "False" olarak yorumlandıktan sonra işleme sokulurlar.
- Ürettikleri sonuçlar ise yine "True" veya "False" olur.
- C'de mantıksal veri türü olmadığından (boolean) mantıksal veriler yerine tam sayılar kullanılır.
- Üretilen sonuç değerine göre:
  - "True" → 1
  - "False" → 0

Mantıksal Operatörler

- Eğer bir sayı mantıksal olarak yorumlanacaksa geçerli kural şudur:
  - 0 → False
  - Sıfır dışındaki negatif veya pozitif sayılar → True
- Örneğin:
  - -11 → True
  - 0 → False
  - 99 → True

Mantıksal Operatörler (! → DEĞİL)

- Tek operandı olan değil operatörü Doğru değeri Yanlış değere, Yanlış değeri Doğru değere dönüştürür.

X	! X
True	False
False	True

- Örneğin: a = !6 → 0

Mantıksal Operatörler (&& → AND)

- Operandlardan ikisi de doğru ise doğru, diğer durumlarda yanlış üretir.

X	Y	X && Y
Yanlış	Yanlış	Yanlış
Yanlış	Doğru	Yanlış
Doğru	Yanlış	Yanlış
Doğru	Doğru	Doğru

- Örneğin: a = !6 → 0

Mantıksal Operatörler (&& → AND)

- AND operatörünün önce sol tarafındaki işlemler yapılır.
- Eğer sol taraf Yanlış (false) ise sağ tarafındaki işlemler yapılmaz.
- Örneğin:
  - a = 4 && 0 → a = 0
  - b = 10 && -4 → b = 1

Mantıksal Operatörler (|| → OR)

- Operandlardan her hangi birisi doğru ise doğru, her ikisi de yanlış ise yanlış üretir.

X	Y	X || Y
Yanlış	Yanlış	Yanlış
Yanlış	Doğru	Doğru
Doğru	Yanlış	Doğru
Doğru	Doğru	Doğru

Mantıksal Operatörler (|| → OR)

- OR operatörünün önce sol tarafındaki işlemler yapılır.
- Eğer sol taraf Doğru (true) ise sağ tarafındaki işlemler yapılmaz.
- Örn:
  - a = 3 || 0 → a = 1
  - b = 0 || -30 → b = 1

Operatörlerde Öncelik Sırası

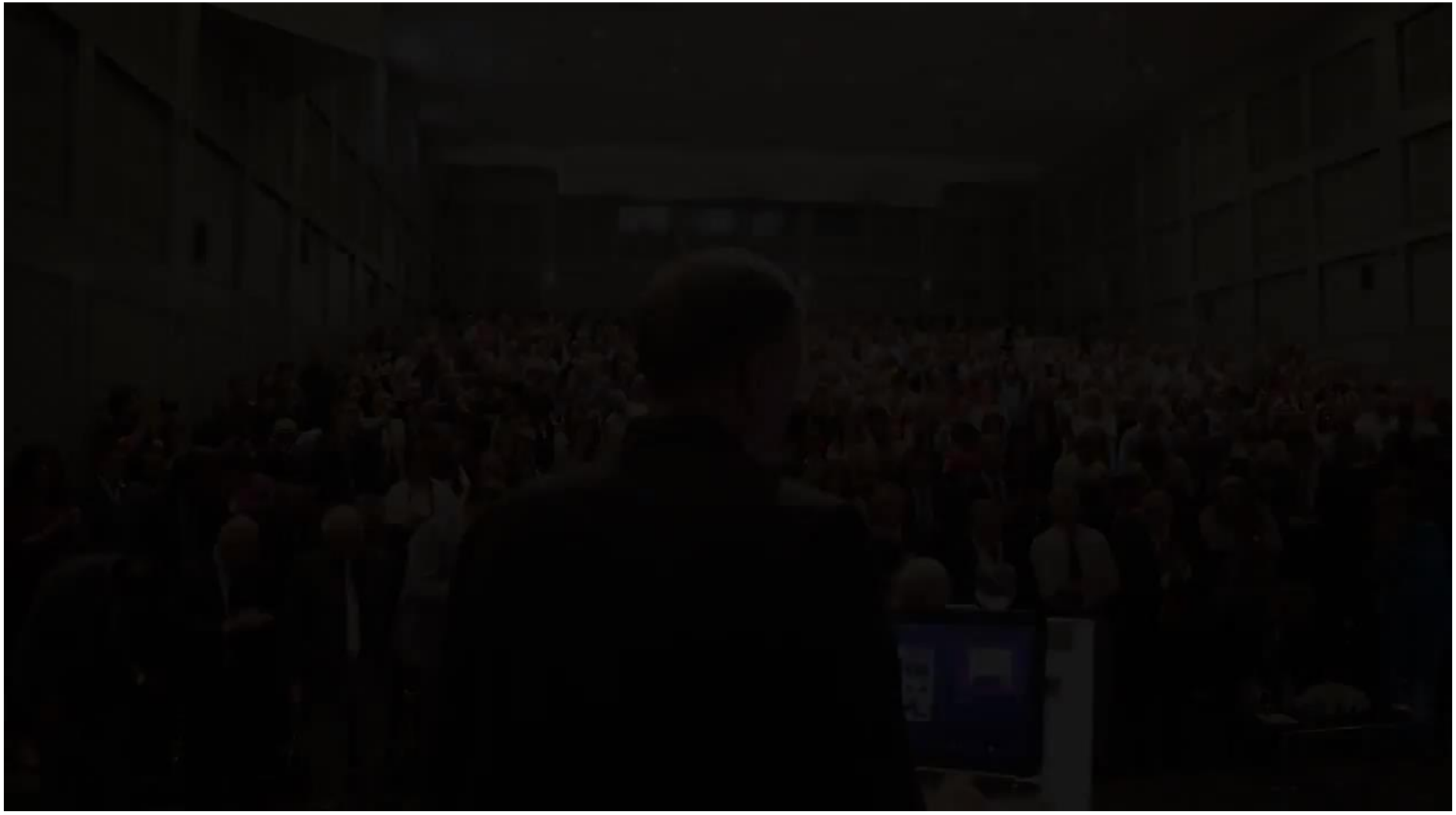
		YÜKSEK ÖNCELİK	
( )	Soldan sağa		Öncelik op.
! ++ --	Sağdan sola		Aritmetik op.
* / %	Soldan sağa		
+ -	Soldan sağa		
> >= < <=	Soldan sağa		İlişkisel op.
== !=	Soldan sağa		
&&	Soldan sağa		Mantıksal op.
||	Soldan sağa		
=	Sağdan sola	**DÜŞÜK ÖNCELİK**	Atama op.

Hatırlayalım, ifadeleri parantez ile ayırmak en iyi yöntemdir.

Operatörlerde Örnek İşlemler

- Örn1:
  - a= 15;
  - x = a >= 10 && a <= 20;
  - Burada x = 1 olur.
- Örn2:
  - a= 20;
  - b= 10;
  - y = a + b >= 20 || a – b <= 10;
  - Burada y = 1 olur.

- Örn3:
  - a= 5;
  - b= 0;
  - y = a || b && a && b
  - Burada y = 1 olur



Kaynaklar

- Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- J. G. Brookshear, “Computer Science: An Overview 10th Ed.”, Addison Wisley, 2009.
- Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- Paul J. Deitel, “C How to Program”, Harvey Deitel.
- Bayram AKGÜL, C Programlama Ders notları