PHP ile İnternet Programlama

2. BÖLÜM: PHP Dilinin Temelleri (a)

Güncelleyen: İdris Kahraman

PHP Sözdizimi Yapısı

Asıl olarak bir dilbilim terimi olan sözdizimi (syntax) kelimesi, programlama dillerinde de benzer bir anlamda kullanılmaktadır. Bir programlama dilinin sözdizimi kuralları, fonksiyonlar, noktalama işaretleri, parantezler ya da operatörler gibi dile ait bütün bileşenlerin, nasıl bir araya gelerek bir program kodunu oluşturacaklarını belirleyen kurallar olarak tanımlanabilir.

Örneğin PHP dilinde bütün anlamlı program deyimlerinin noktalı virgül işareti ile sonlandırılması zorunluluğu temel bir sözdizimi kuralıdır.

Burada vurgulanması gereken önemli bir nokta, satırların değil, deyimlerin sonuna noktalı virgül konulmasıdır. Çünkü PHP çözümleyicisi kaynak kodları satır satır değil, ayıraç sembolü olarak noktalı virgül işaretini kullanarak yorumlar.

PHP Sözdizimi Yapısı

PHP kodları çoğunlukla HTML kodları ile aynı sayfada yer aldığından, PHP çözümleyicisinin görev alanının bir kod bütününde nerede başladığı ve nerede sona erdiğinin belirlenmesi amacıyla <?php … ?> etiket (tag) sistemi kullanılır. Örneğin,

```
<?php
    echo "Merhaba Dünya!";
?>
```

kodu doğrudan PHP çözümleyicisi tarafından yorumlanarak çalıştırılabilecek bir koddur. Kodun işlevi ise ekrana Merhaba Dünya metnini yazdırmaktır.

PHP Sözdizimi Yapısı

Aynı işleve sahip, ancak bu defa HTML kodu içersine “gömülmüş” durumda olan bir başka PHP kodu aşağıda gösterilmiştir.

```
<html>
<body>
<?php
    echo "Merhaba Dünya!";
?>
</body>
</html>
```

PHP kodlarının HTML kodlarıyla birlikte kullanıldığı bu gibi durumlarda, PHP deyimlerinin web tarayıcısı tarafından çözümlenerek çalıştırılabilmesi için bu kodu içeren sayfanın “php” uzantısına sahip olması gerektiğini unutmayalım.

PHP Sözdizimi Yapısı

Ekrana yazdırmak istediğimiz ifadeler için, echo ya da print deyimlerini kullanırız. Bu deyimlerden sonra çift tırnak sembolü içersinde yazacağımız ifadeler, bazı özel durumların dışında, doğrudan yazıldığı şekliyle ekrana basılacaktır. Örneğin,

```
<html>
<body>
<?php
    print "Merhaba Dünya!";
?>
</body>
</html>
```

kodunun çıktısı,

```
Merhaba Dünya!
```

biçiminde olacaktır.

PHP Sözdizimi Yapısı

ECHO ve PRINT Arasındaki Fark Nedir?

İşlevsel anlamda bu iki deyim arasında çok fazla bir fark bulunmamaktadır. Her iki deyim de ekrana “string” temelli bilgileri yazmak için kullanılır.

Print deyimi, Echo deyiminden farklı olarak bir fonksiyon gibi çalışır ve bir geri dönüş değeri üretir. Bu sebeple çok büyük verilerin ekrana yazdırılmasında Echo deyimine göre daha yavaş çalıştığı söylenir. Ancak bu hız farkı, kayda değer bir yavaşlamaya yol açmadığından çok fazla önemsenmez.

Diğer bir fark, Echo deyimi ile yapılabilen ardışık ifadelerin virgül ile yazdırılması işleminin, Print deyimi kullanılarak yapılamamasıdır.

PHP Sözdizimi Yapısı

ECHO ve PRINT Arasındaki Fark Nedir?

```
<?php
    print "Deneme", " yayını";
?>
```

kodu çözümleme hatasına yol açarken, aynı kodu Echo deyimi ile çalıştırdığımızda sonuç alabiliriz.

PHP Sözdizimi Yapısı

ECHO ve PRINT Arasındaki Fark Nedir?

Print deyimi yazdırılacak ifadeyi argüman olarak kabul eder ve her zaman “1” değerini döndürür:

```
<?php
    echo print "Deneme";
?>
```

```
Deneme1
```

PHP Sözdizimi Yapısı

```
<?php
    echo "Merhaba ";
    echo "Dünya!";
?>
```

koduna ait ekran çıktısı;

```
Merhaba Dünya!
```

biçiminde olur.

Görüldüğü gibi farklı echo deyimleri kullanılarak yazılsalar da, ifadeler aynı satıra yazılmaktadır. Eğer farklı satırlarda görmek istiyorsak, bu durumda yazacağımız koda küçük bir ek yapmamız gerekir.

PHP Sözdizimi Yapısı

```
<?php
    echo "Merhaba";
    echo "<br>";
    echo "Dünya!";
?>
```

koduna ait ekran çıktısı;

```
Merhaba
Dünya!
```

biçiminde olacaktır.

PHP Sözdizimi Yapısı

Son örnekte yer alan echo "
"; ifadesi, küçük ama önemli bir bilgiyi içermektedir. Bu echo deyimi ile yazdırdığımız "
" ifadesi ekrana herhangi bir şey yazmamış, bunun yerine birinci ve üçüncü echo deyimlerinin yazdıkları ifadelerin arasında satırbaşı yapma işlevini sağlamıştır.
 İfadesi aslında bir HTML etiketidir ve işlevi de tam olarak budur.

Dolayısıyla internet tarayıcısı echo deyimi ile ekrana yazılacak ifadeleri denetlemekte, bu ifadeleri HTML kodu olarak yorumladığı takdirde doğrudan ekrana yazmak yerine çalıştırmakta ve sonucunu görüntülemektedir.

PHP Sözdizimi Yapısı

Echo Deyimi kullanılarak gerçekleştirilmiş, PHP içersinden çalıştırılan HTML kodlarına verilebilecek başka bir örnek;

```
<?php
    echo "<p><strong>Merhaba Dünya!</strong></p>";
?>
```

Bu durumda ekrana yazılacak metin kalın (strong ya da bold) türde olacaktır.

Bu yöntemde karşılaşabileceğimiz bir problem, çift tırnak işaretlerinin kullanımında yaşanacaktır. Şöyle ki çift tırnak, bir yandan echo deyiminin etki alanının belirlenmesinde kullanılırken, diğer yandan bir çok HTML deyiminde değerlerin belirlenmesinde kullanılmaktadır. Bu gibi durumlarda PHP dili için özel karakterlerden birisi olan \ (ters bölü) işaretini kullanılır.

Bir sonraki örneğimiz, bu özelliğin kullanımına ilişkindir.

PHP Sözdizimi Yapısı

```
<?php
    echo "<font size=\"5\">SINAVA GİRECEKLER</font>";
    echo "<p>";     echo "1. Kemal SÜZER<br>";
    echo "2. Melek ÇALIŞ<br>";
    echo "3. Ercan METİN<br>";
    echo "4. Meral YILMAZ"; echo "</p>";
?>
```

SINAVA GİRECEKLER

1. Kemal SÜZER
2. Melek ÇALIŞ
3. Ercan METİN
4. Meral YILMAZ

PHP Sözdizimi Yapısı

Görüldüğü gibi burada, ilk echo deyimi ile yazdırılan ifade bir HTML deyimidir. Bu deyim içersinde, “size“ parametresinin değerinin belirlenmesi için çift tırnak işareti bulundurmaktadır. Bu çift tırnak işaretlerinin, echo deyimine ait olan ve PHP tarafından dikkate alınacak olan (kodda kırmızı renkte gösterilmişlerdir) çift tırnaklardan ayırt edilebilmeleri için \ işareti kullanılmıştır.

PHP Sözdizimi Yapısı

Alternatif olarak echo ya da print deyimleri içersindeki HTML kodlarında tek tırnak işareti de kullanılabilir. HTML yorumlayıcısı, çift tırnak işareti yerine bu işareti de kabul etmektedir. Bu durumda \ işaretinin kullanımına gerek kalmayacaktır.

Aşağıdaki kod, bir önceki ile aynı sonucu üretecektir.

```
<?php
    echo "<font size='5'>SINAVA GİRECEKLER</font>";
    echo "<p>";     echo "1. Kemal SÜZER<br>";
    echo "2. Melek ÇALIŞ<br>";
    echo "3. Ercan METİN<br>";
    echo "4. Meral YILMAZ"; echo "</p>";
?>
```

Açıklama Metinleri

Kod içersinde, PHP yorumlayıcısı tarafından dikkate alınmayacak, açıklama yapmak amaçlı olarak açıklama metinleri yer alabilir. Bu açıklamalar program çıktısında görünmezler.

Tek satırlı ya da çok satırlı olarak açıklama metni eklenebilir. Aşağıda, koda farklı şekillerde eklenmiş açıklama metinleri verilmiştir.

```
<?php
    echo 'PHP '; // Bu tek satırlık açıklamadır.
    /* Bu bir çok-satırlı
       açıklamadır. */
    echo 'öğreniyorum!';
    # Bu da tek satırlık açıklamadır.
?>
```

Veri Türleri

Üç temel veri türünü, sayısal olmayan string-tabanlı türler (text, string, char gibi), sayısal türler (integer, double gibi) ve diziler (array) şeklinde sıralayabiliriz.

Diziler, özel bir tanımlama biçimine sahip olduklarından diğer veri türlerinden kolaylıkla ayırt edilebilirler.

String-tabanlı türler ise, aritmetik işlemlerde kullanılmayan türlerdir ve sayısal türlerle aralarındaki en önemli fark, bu türdeki değerlerin “ “ ya da ‘ ‘ (çift ya da tek tırnak) işreti arasında yer almalarıdır.

Veri Türleri

Örneğin,

2

değeri sayısal bir veri türü iken,

“2”

değeri 1 baytlık bir string olacaktır. Dolayısıyla veri türü dönüştürme işlemi uygulanmadan aritmetik işlemlerde kullanılamaz.

Değişkenler ve Sabitler

Programların çalıştıkları sırada kullandıkları çeşitli değerleri bellekte saklama zorunlulukları vardır. Bu değerler kullanıcılar tarafından girilen değerler olabilecekleri gibi, doğrudan programın kendi ürettiği sonuçlar da olabilir.

Değerlerin saklandıkları temel dil bileşenleri değişkenler ve sabitlerdir. Birbirlerine çok yakın bir işleve sahip olan bu bileşenler arasındaki temel fark, değişkenlerin taşıdıkları değerlerin programın çalışması süresince dinamik olarak değişebilmesi, buna karşılık sabitlerin aldıkları ilk değerlerini programın çalışması süresince korumalarıdır.

Değişkenler ve Sabitler

Değişkenler

Bir değişken için iki önemli özellik söz konusudur: Değişkenin adı ve sakladığı verinin türü.

PHP dili, güçlü-türsel diller (strongly-typed languages) sınıfına girmediğinden, değişkenlerin bu dillerde olduğu gibi önceden tanımlanmasına (deklare edilmesine) gerek yoktur. Dolayısıyla bir değişken, doğrudan ilk değer ataması ile otomatik olarak deklare edilmiş olacaktır.

PHP dilinde değişkenleri diğer dil öğelerinden ayıran simge $ (dolar) işaretidir. Her değişken adı, $ işareti ile başlar ve "değişken isimlendirme kuralları" gözetilerek isimlendirilir.

Değişkenler ve Sabitler

Değişkenler

Değişkenleri isimlendirirken dikkat etmemiz gereken kuralları aşağıda verilmiştir:

Bir değişken ismi, bir harf ya da _ (alt tire) sembolü ile başlayabilir.

Bir değişken ismi sadece alfa-nümerik karakterleri (a-z, A-Z, 0-9) ve alt tireyi (_) içerebilir. Türkçe harfler (ç,Ç,ğ,Ğ,ı,İ,ö,Ö,ü,Ü) bunlara dahil değildir.

Değişken isimleri boşluk içeremezler. Ayırma gerektiğinde alt tire (bolum_kodu gibi) ya da büyük harfler (bolumKodu gibi) kullanılabilir.

Değişkenler ve Sabitler

Değişkenler

Bu durumda PHP dilinde bir değişkenin tanımlanması ve değer ataması, genel olarak aşağıdaki şekilde olacaktır:

$degisken_adi = deger;

Daha somut bir örnek aşağıda verilmiştir:

```
<?php
    $p="Merhaba!";
    $sayi=7;
?>
```

Değişkenler ve Sabitler

Değişkenler

Burada $p string türünde bir değişken olarak tanımlanmıştır. $sayi ise tamsayı türünde bir değişken olarak kullanılmıştır.

“=“ sembolü ile yapılan bu işleme “değer atama/aktarma” adı verilir. Değer atama işlemi = sembolünün sağından soluna doğru yapılır. Dolayısıyla eşitliğin solunda herzaman bir değişken sembolü bulunmalıdır.

Değişkenler ve Sabitler

Değişkenler

Değer aktarmalarda, aktarılan değerler otomatik olarak ekranda görüntülenmezler. Bunun için yine echo deyiminden yararlanabiliriz:

```
<?php
    $p="Merhaba!";
    $sayi=7;
    echo $p;
    echo $sayi;
?>
```

```
Merhaba!7
```

Değişkenler ve Sabitler

Değişkenler

Değişkenlerin içerdikleri değerler ile sabit metinleri bir arada görebiliriz.

```
<?php
    $metin="Merhaba!";
    $sayi=7;
    echo "Metin:$metin Sayı:$sayi";
?>
```

```
Metin:Merhaba! Sayı:7
```

Değişkenler ve Sabitler

Değişkenler

Bu örneğimizde, sayi1 ve sayi2 değişkenlerinin taşıdıkları değerler üçüncü bir değişken (sayi3) yardımı ile yer değiştirmektedirler.

```
<?php
    $sayi1 = 4;
    $sayi2 = 10;
    $sayi3=$sayi1;
    $sayi1=$sayi2;
    $sayi2=$sayi3;
    echo "Sayı-1: $sayi1 <br> Sayı-2: $sayi2";
?>
```

```
Sayı-1: 10
Sayı-2: 4
```

Değişkenler ve Sabitler

Değişkenler

Aşağıda, bir önceki örnek ile aynı işlevi üçüncü bir değişken kullanmadan gerçekleştiren kod yer almaktadır.

```
<?php
    $sayi1 = 4;
    $sayi2 = 10;
    $sayi2=$sayi1+$sayi2;
    $sayi1=$sayi2-$sayi1;
    $sayi2=$sayi2-$sayi1;
    echo "Sayı-1: $sayi1 <br> Sayı-2: $sayi2";
?>
```

Değişkenler ve Sabitler

Değişkenler

Aşağıdaki örnek ise değer aktarma sırasında string türde bir tanımlama yapılmasına karşın, aritmetik işleme sokulduğunda PHP yorumlayıcısı tarafından otomatik olarak sayısal türe dönüştürülen bir değişkenin hikayesini anlatmaktadır.

```
<?php
    $sayi1 = 3;
    $sayi2 = "5";
   $sayi3 = $sayi1 + $sayi2;
   echo "$sayi1 + $sayi2 = $sayi3";
?>
```

```
3 + 5 = 8
```

Değişkenler ve Sabitler

Değişkenler

PHP dilinde oldukça kullanışlı özelliklerden birisi de, değişkenlerin taşıdıkları değerler kullanarak yeni değişken isimlerinin oluşturulabilmesidir.

```
<?php
    $a = 'merhaba';
    $$a = 'dünya';
    echo $merhaba;
?>
```

```
dünya
```

Değişkenler ve Sabitler

Değişkenler

Burada, $a değişkeninin değeri olan “merhaba” metni kullanılarak yeni bir $merhaba değişkeni tanımlanmış, bu değişkene değer olarak da “dünya” metni aktarılmıştır. Bu işlem, ikinci bir $ simgesi kullanılarak gerçekleştirilmiştir.

Değişkenler ve Sabitler

Değişkenler

Aynı çıktıyı aşağıdaki kod ile elde etmek mümkündür:

```
<?php
    $a = 'merhaba';
    $$a = 'dünya';
    echo "${$a}";
?>
```

Değişkenler ve Sabitler

Değişkenler

Bu durumda aşağıdaki kodun ekran çıktısı;

```
<?php
    $a = 'merhaba';
    $$a = 'dünya';
    echo "$a ${$a}";
?>
```

```
merhaba dünya
```

biçiminde olacaktır.

Değişkenler ve Sabitler

Sabitler

Değişkenlere benzerler, ancak sabitlerin taşıdıkları değerler, programın çalışması süresince aynı kalır. “Define” bildirimi kullanılarak tanımlanırlar.

Aşağıda sabit tanımlama ile ilgili örnek bir kod parçası verilmiştir:

```
define("gelir_vergisi", 34.29);
define("damga_vergisi", 13.26);
define("saglik_kesintisi", 210.64);
define("emeklilik_kesintisi", 280.06);
```

Değişkenler ve Sabitler

Sabitler

Sabitler, değişkenlerden farklı olarak isimlerinin başlarında $ simgesi taşımazlar. Yukarıda tanımlanan sabitlerin nasıl kullanılacağına ilişkin örnek kod parçası aşağıda verilmiştir.

```
$toplam_kesinti = gelir_vergisi + damga_vergisi
    + saglik_kesintisi + emeklilik_kesintisi;
```

Operatörler

Operatörler, değerler üzerinde ikili işlemlerin gerçekleştirilmesinde kullanılırlar. PHP dilindeki operatörleri genel olarak dört kategori altında toplayabiliriz:

1. Aritmetik Operatörler
2. Değer Atama Operatörleri
3. Karşılaştırma Operatörleri
4. Mantıksal Operatörler

Şimdi bunları ayrı ayrı ele alalım.

Operatörler

Aritmetik Operatörler

Değişkenler ve sabit değerler üzerinde her türlü aritmetik işlem, aritmetik operatörler kullanılarak gerçekleştirilir. PHP dilinde “. “ (nokta) sembolü ise metinsel birleştirme işlevine sahiptir.

Operatörler

Operatör	Tanımı	Örnek Kod Parçası	Sonuç
+	Toplama	$x=2; $sonuc=$x+2;	 4
-	Çıkartma	$x=2; $sonuc=$x-1;	 1
*	Çarpma	$x=4; $sonuc=$x*5;	 20
/	Bölme	$x=15; $sonuc1=$x/5; $sonuc2=$x/2;	 3 7.5
%	Mod (Bölmede kalan)	$sonuc1=5%2; $sonuc2=10%8; $sonuc3=10%2;	1 2 0
++	Artırma	$x=7; $x++;	 $x=8
--	Azaltma	$x=10; $x--;	 $x=9
.	Metinsel birleştirme	$x1="LAB"; $x2="101"; $sonuc=$x1.$x2;	 “LAB101”

Operatörler

Değer Atama Operatörleri

Temel değer atama operatörü = simgesidir. Atama, bu simgenin sağından soluna doğru gerçekleşir.

Diğer operatörler = operatörünün aritmetik operatörlerle birlikte kullanılması ile türetilmişlerdir. Bunlar, bir aritmetik işlem ve sonrasında gerçekleştirilecek değer atama işlemini birleştirerek tek seferlik bir atama işlem haline getirirler.

Operatörler

Operatör	Örnek Kod Parçası	Eşdeğer İşlem
=	$x=$y;	
+=	$x+=$y;	$x=$x+$y;
-=	$x-=$y;	$x=$x-$y;
*=	$x*=$y;	$x=$x*$y;
/=	$x/=$y;	$x=$x/$y;
.=	$x.=$y;	$x=$x. $y;
%=	$x%=$y;	$x=$x%$y;

Operatörler

Aşağıda, değişken oluşturma ve aritmetik işlem yaparak değer atama konularını birlikte içeren güzel bir örnek kod verilmiştir.

```
<?php
    $isim = "sayi";
    $n1 = "1";
    $n2 = "2";
    ${$isim.$n1} = 12;
    ${$isim.$n2} = 5;
    $sayi1%=$sayi2;
    echo "$sayi1";
?>
```

Bu programın sonucu ne olur?

Operatörler

Karşılaştırma Operatörleri

Karşılaştırma operatörleri, ilerleyen kesimlerde ele alacağımız koşullu ifadelerde, koşulun oluşturulmasında kullanılmaktadır. Bu operatörler kullanılarak gerçekleştirilecek bir karşılaştırma işlemi, true (doğru) ya da false (yanlış) değerini üretir.

Aşağıdaki tabloda PHP dilinde kullanılan karşılaştırma operatörleri örnekler verilerek açıklanmıştır.

Operatörler

Operatör	Tanımı	Örnek
==	Eşittir	3==2 *False değer üretir.*
!=	Eşit değildir	3!=2 *True değer üretir.*
>	Büyüktür	3>2 *True değer üretir.*
<	Küçüktür	3<2 *False değer üretir.*
>=	Büyük ya da eşittir	3>=2 *True değer üretir.*
<=	Küçük ya da eşittir	3<=2 *False değer üretir.*

Operatörler

Mantıksal Operatörler

Mantıksal operatörler de karşılaştırma operatörleri gibi koşullu ifadelerde bileşik koşulların oluşturulmasında kullanılırlar.

Aşağıdaki tabloda PHP dilinde kullanılan mantıksal operatörler, örnekler verilerek açıklanmıştır.

Operatörler

Operatör	Tanımı	Örnek
&&	ve	$x=8 $y=5 ($x < 10 && $y > 1) *True değer üretir.*
||	veya	$x=8 $y=5 ($x==5 || $y==5) *True değer üretir.*
!	değil	$x=8 $y=8 !($x==$y) *False değer üretir.*

Diziler

Diziler, birden çok sayıda verinin tek bir değişken ile temsil edilmeleri gerektiğinde kullanılırlar. Dizi içersinde istenen veriye bir indis değişkeni kullanılarak ulaşılır.

İleriki bölümlerde ele alacağımız döngü yapıları ile kullanıldıklarında, gereksiz kod yazımını da önemli ölçüde azaltan diziler (array), tek boyutlu ve çok boyutlu olarak sınıflandırılabilirler.

Diziler

Tek Boyutlu Diziler

Konuya bir örnekle başlayalım:

```
<?php
    $arabalar1="Honda";
    $arabalar2="Volvo";
    $arabalar3="BMW";
    $arabalar4="Toyota";
?>
```

Sadece dört tane araba markasını saklamak istediğimizden, dört tane farklı değişken kullandık. Peki 400 farklı markayı saklamak isteseydik?

Diziler

Tek Boyutlu Diziler

Aynı örneği, dizi kullanarak yapalım:

```
<?php
    $arabalar=array("Honda","Volvo","BMW","Toyo
ta");
?>
```

Bu defa tanımlamış olduğumuz tek bir değişken olan $arabalar'dır. Ancak bu değişken, "array" bildirimi kullanılarak oluşturulmuş bir dizidir.

Diziler

Tek Boyutlu Diziler

Aynı tanımlamayı farklı bir şekilde de yapabilirdik:

```
<?php
    $arabalar[0]="Honda";
    $arabalar[1]="Volvo";
    $arabalar[2]="BMW";
    $arabalar[3]="Toyota";
?>
```

Bu tanımlamada, dizinin indisi adı verilen köşeli parantezlerin içerisinde yer alan değerleri kendimiz belirledik.

Diziler

Tek Boyutlu Diziler

Her iki tanımlama için de aşağıdaki kod parçası,

```
echo "$arabalar[0] ve $arabalar[3] Japon otomobil markalarıdır.";
```

aynı ekran çıktısını verecektir:

```
Honda ve Toyota Japon otomobil markalarıdır.
```

Akış Kontrol Deyimleri

PHP programlarının normal akışı, yukarıdan aşağıya doğru doğrusal olarak gerçekleşmektedir. Ancak programcı tarafından bu akışın doğrusallığı zaman zaman sıçramalı ya da döngüsel yapılarla değiştirilebilir. Bu değişikliği gerçekleştiren deyimlere akış kontrol deyimleri adı verilir.

Akış kontrol deyimlerini genel olarak koşullu ifadeler ve döngüler şeklinde iki kategoride sınıflayabiliriz.

Akış Kontrol Deyimleri

Koşullu İfadeler

Bilgisayar programları için önemli bir özellik olan “karar verme” olgusu, en basit şekliyle bilgisayarın belirli bir durum karşısında, önüne gelen seçeneklerden en uygun olanını seçerek ona uygun olarak davranması şeklinde açıklanabilir.

Diğer bir deyişle bir program için karar verme durumu, program akışının “koşul” adı verilen ve Boolean, yani True (doğru) ya da False (yanlış) değer alan ifadenin, aldığı değere göre yönlendirilmesi biçiminde gerçekleştirilir.

Akış Kontrol Deyimleri

Koşullu İfadeler

En temel koşullu ifade biçimi, Türkçede “eğer” anlamına gelen “if (koşul) …” yapısıdır.

Bununla birlikte, gelişmiş programlama dillerinin hemen hemen hepsinde bulunan diğer bir koşul yapısı ise bir koşula ait çok sayıda durumun birlikte değerlendirildiği “switch…case” yapısıdır.

Akış Kontrol Deyimleri

Koşullu İfadeler – If

En temel koşul yapısı

if (koşul) {a}

şeklindedir. Burada {a} kısmına, koşul sağlandığında gerçekleştirilecek deyim (ya da deyimler) yazılır.

Sonraki slaytta bununla ilgili bir örnek program ve ekran çıktısı yer almaktadır.

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Else

Aksi halin söz konusu olduğu durumlarda ise

if (koşul) {a} else {b}

ifadesi kullanılır. Bu durumda koşulun aksinin gerçekleştiği, yani koşulun aldığı değerin değilinin elde edildiği durumlarda else ifadesinden sonra yer alan {b} deyimi (ya da deyimleri) çalıştırılacaktır.

Örnek program sonraki slaytta verilmiştir.

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Else

```
<?php
    $sayi=15;
    if ($sayi%2==0) echo "$sayi bir çift sayıdır.";
    else echo "$sayi bir tek sayıdır.";
?>
```

```
15 bir tek sayıdır.
```

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Else

Bu programla aynı sonucu, aşağıdaki kod ile de alabiliriz:

```
<?php
    $sayi=15;
    if ($sayi%2==0) echo "$sayi bir çift sayıdır.";
    if (!$sayi%2==0) echo "$sayi bir tek sayıdır.";
?>
```

Ancak bu durumda kodu gereksiz yere uzatmış oluruz. Unutmayın, en iyi program, yapılmak isteneni en az kod yazarak gerçekleştiren programdır.

Bir koşul ve onun eksi halinin söz konusu olduğu durumlarda, yukarıdaki gibi ilk koşulun değili olan ikinci bir koşul yazmak yerine, else deyimini kullanmak daha yerinde olacaktır.

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Else

Koşul ya da aksi durumun doğrulanması durumunda gerçekleştirilecek işlem sayısı birden fazla olabilir. Bu gibi durumlarda, programlama dillerinde bloklama etiketleri kullanılır.

PHP dilinde bloklama işlemlerinde { ve } (açık ve kapalı küme parantezleri) kullanılır.

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Else

Örnek program için şöyle bir senaryo geliştirelim:

Bir mağaza, müşterilerine iki türde indirim uygulamaktadır. Bunlardan ilki, hafta sonu indirimidir. Yani alışveriş edilen gün Cumartesi ya da Pazar ise toplam alışveriş tutarına %10 indirim uygulanmaktadır.

İkincisi ise, 150 TL ve üstü alışverişlerde uygulanan %20 indirimdir.

Her iki duruma da giren alışverişler, iki indirimden de faydalanabilmektedir.

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Else

```
<?php
    $alisverisTutari=200;
    $gun=date("D");
    if ($gun=="Sat" || $gun=="Sun")
    {
        $alisverisTutari-=10*$alisverisTutari/100;
        echo "Haftasonu indirimi uygulanmıştır.
              Yeni tutar: $alisverisTutari";
    }
    if ($alisverisTutari>=150)
    {
        $alisverisTutari-=20*$alisverisTutari/100;
        echo "150 TL üstü indirimi uygulanmıştır.
              Yeni tutar: $alisverisTutari";
    }
?>
```

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Elseif/Else

If deyiminin en kapsamlı biçimi, yapıya “aksi halde eğer” anlamını taşıyan “Elseif” bölümünün eklendiği biçimdir. Koşulun aksi halinde başka alt koşulların bulunması durumunda kullanılır.

Genel yapısı aşağıdaki gibidir:

If (koşul) {a}
Elseif (koşul_1) {a_1}
Elseif (koşul_2) {a_2}
…
Elseif (koşul_n) {a_n}
Else {b}

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Elseif/Else

Bu yapıda, ilk If deyimine ait koşul sağlanırsa diğer durumlara bakılmaksızın sadece {a} deyimi/deyimleri çalıştırılır ve program akışı koşul yapısının sonundan devam eder.

İlk koşul sağlanmazsa sırasıyla koşul_1, koşul_2, …, koşul_n koşulları denetlenir. Bunlardan hangisi sağlanırsa, ilgili deyim/deyimler çalıştırılır.

Hiçbir koşulun sağlanmadığı durumda ise, Else deyiminden sonra yer alan {b} deyimi/deyimleri çalıştırılır.

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Elseif/Else

Örnek program için şöyle bir senaryo geliştirelim:

Kullanıcı tarafından girilecek bir notun, aşağıdaki tabloya göre karşılık gelen harf notu hesaplanacak ve kullanıcıya bildirilecektir.

Not Aralığı	Karşılık Gelen Harf
0-19	FF
20-29	FD
30-39	DD
40-49	DC
50-59	CC
60-69	CB
70-79	BB
80-89	BA
90-100	AA

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Elseif/Else

```
<?php

/* Nota karşılık gelen harf notu belirleniyor. */

    $Not=65;
    if ($Not<=19) echo "$Not = FF";
    elseif ($Not<=29) echo "$Not = FD";
    elseif ($Not<=39) echo "$Not = DD";
    elseif ($Not<=49) echo "$Not = DC";
    elseif ($Not<=59) echo "$Not = CC";
    elseif ($Not<=69) echo "$Not = CB";
    elseif ($Not<=79) echo "$Not = BB";
    elseif ($Not<=89) echo "$Not = BA";
    elseif ($Not<=100) echo "$Not = AA";
    else echo "$Not notuna karşılık gelen harf notu
yoktur.";
?>
```

Akış Kontrol Deyimleri

Koşullu İfadeler – If/Elseif/Else

Bu örneğimizde de kişinin cinsiyetine göre boyu ortalama boy ile karşılaştırılmaktadır.

```
<?php
    $cinsiyet="e"; $boy=180;
    if($cinsiyet=="k")
    {if ($boy<160)
          {echo "Boyunuz ortalamanın altındadır."; }
          elseif ($boy>160)
                echo "Boyunuz ortalamanın üzerindedir.";
          else
                echo "Boyunuz tam ortalamadadır."; }
    elseif($cinsiyet=="e")
    {if ($boy<170) echo "Boyunuz ortalamanın altındadır.";
          elseif ($boy>170)
                echo "Boyunuz ortalamanın üzerindedir.";
          else echo "Boyunuz tam ortalamadadır.";
    }else echo "Cinsiyet bilgisi hatalı girildi..."; ?>
```

Akış Kontrol Deyimleri

Koşullu İfadeler – Switch/Case

Aynı ifadenin, aldığı farklı değerlere karşılık gelen durumlarının belirlendiği koşullu ifade biçimi switch/case yapısıdır.

Aslında bu yapı kullanılarak gerçekleştirilen akış kontrolü, bir dizi if-elseif ifadesi kullanılarak da gerçekleştirilebilir. Ancak bu durum gereksiz kod kalabalığına ve karmaşıklığına yol açacaktır.

Akış Kontrol Deyimleri

Koşullu İfadeler – Switch/Case

İfadenin genel biçimi aşağıdaki gibidir:

```
switch (ifade)
{
case değer1:
  ifade=değer1 olduğunda işletilecek kod;
  break;
case değer2:
 ifade=değer2 olduğunda işletilecek kod;
  break;
default:  ifade, değer1 ve değer2'nin her ikisinden de farklı
olduğunda işletilecek kod;  }
```

Akış Kontrol Deyimleri

Koşullu İfadeler – Switch/Case

Örnek olarak 0 ile 9 arasındaki bir sayının Türkçe okunuşunu yazacak PHP kodunu switch/case yapısını kullanarak yazalım.

Akış Kontrol Deyimleri

```
<?php
        $Gelen_Sayi=7;
        switch ($Gelen_Sayi) {
        case "0":
        $sonuc="SIFIR";
        break;
        case "1":
        $sonuc="BİR";
        break;
        case "2":
        $sonuc="İKİ";
        break;
         case "3":
        $sonuc="ÜÇ";
        break;
        case "4":
        $sonuc="DÖRT";
        break;
        case "5":
        $sonuc="BEŞ";
            break;
        case "6":
        $sonuc="ALTI";
            break;
        case "7":
        $sonuc="YEDİ";
            break;
        case "8":
        $sonuc="SEKİZ";
        break;
        case "9":
        $sonuc="DOKUZ";
        break;
    default: $sonuc="Giriş Anlaşılamadı...";}}
echo ($sonuc); ?>
```