PHP ile İnternet Programlama

Prof.Dr. Tolga GÜYER

Gazi Üniversitesi
Gazi Eğitim Fakültesi
Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü

2. BÖLÜM: PHP Dilinin Temelleri (b)

Akış Kontrol Deyimleri

Döngüler

Konuya bir örnekle başlayalım.

Ekranda alt alta 7 defa PHP Öğreniyorum cümlesini aşağıda gösterildiği gibi küçükten büyüğe doğru artan yazıtipi büyüklüğü ile yazdıracağız:

PHP Öğreniyorum!
PHP Öğreniyorum!
PHP Öğreniyorum!
PHP Öğreniyorum!
PHP Öğreniyorum!
PHP Öğreniyorum!
PHP Öğreniyorum!

Akış Kontrol Deyimleri

Döngüler

İstediğimiz görüntüyü verecek kod aşağıdaki gibi olacaktır:

```
<?php
     echo "<font size=1>";
     echo "<br><b>PHP Öğreniyorum!</b></font>";
     echo "<font size=2>";
     echo "<br><b>PHP Öğreniyorum!</b></font>";
     echo "<font size=3>";
     echo "<br><b>PHP Öğreniyorum!</b></font>";
     echo "<font size=4>";
     echo "<br><b>PHP Öğreniyorum!</b></font>";
     echo "<font size=5>";
     echo "<br><b>PHP Öğreniyorum!</b></font>";
     echo "<font size=6>";
     echo "<br><b>PHP Öğreniyorum!</b></font>";
     echo "<font size=7>";
     echo "<br><b>PHP Öğreniyorum!</b></font>";
?>
```

Akış Kontrol Deyimleri

Döngüler

Örneğimizde olduğu gibi belirli bir kural dahilinde tekrarlanan işlemlerde, işlemi gerçekleştiren kodun da hemen hemen tamamının işlem sayısı kadar tekrarlanması zorunluluğu ortaya çıkmaktadır. Bu da, iyi bir programcının uzak durması gereken en önemli problemlerden birisi olan gereksiz kod kalabalığına yol açmaktadır.

Bu gibi durumlar için geliştirilen programatik yapılara döngü adı verilir. Döngüler, belirli bir düzen içinde tekrarlanan kodların sadece bir defa yazılmasını sağlarlar.

Akış Kontrol Deyimleri

Döngüler

Genel olarak döngüleri iki başlık altında sınıflandırabiliriz. Bunlardan ilki, döngü tarafından değeri otomatik olarak artırılan bir sayaç değişkeninin bulunduğu sayaçlı döngülerdir.

Diğer tür olan koşullu döngülerde ise döngünün sonlanması için belirli bir koşulun sağlanması söz konusudur.

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler

Belirli bir indis değişkenine bağlı olarak işlem yapan döngü türleridir. PHP'de iki tür sayaçlı döngü vardır: For döngüsü ve Foreach döngüsü.

For döngüsünde indis değişkeninin doğrudan döngünün tanımlama bloğu içersinde belirtilmesi gerekir.

Foreach döngüsünde ise indis değişkeninin yerini bir dizi alır. Dizinin kendi yapısında bir indis değeri zaten tanımlı olduğundan, Foreach yapısında sayaç olarak bu indis değerinin kullanıldığı söylenebilir.

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

Genel yapısı aşağıdaki gibidir:

```
For (Başlangıç değeri; Sınır koşulu; Artış miktarı)
 {
        Döngü tarafından çalıştırılacak kod;
 }
```

Başlangıç değeri çoğunlukla sayaç işlevini görecek değişkene ilk değerinin atanması şeklinde belirlenir.

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

Sınır koşulu, her döngü adımında yeniden hesaplanan bir Boolean değer üretir. Sonuç True olduğu sürece döngü çalışır, aksi halde sonlanır. Genellikle sayaç değişkeni için bir üst sınır değerinin atanması şeklinde belirlenir.

Artış miktarı sayaç değişkeninin nasıl artacağının (ya da azalacağının) belirlendiği parametredir.

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

Örnek olarak, daha önce yaptığımız “PHP Öğreniyorum!” uygulamasını bu kez For döngüsünü kullanarak gerçekleştirelim.

```
<?php
for ($i=1; $i<=7; $i++)
  {
     echo "<font size=$i>";
     echo "<br><b>PHP Öğreniyorum!</b></font>";
  }
?>
```

Sonuç yine aynı olacaktır. Ancak kodun ilk hali ile karşılaştırıldığında kod kalabalığının oldukça azaldığı görülmektedir.

Burada sayaç değişkeni olarak $i belirlenmiş, bu değer yazıtipi büyüklüğünü 1’den 7’ye kadar artırmak için kullanılmıştır.

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

Şimdi aşağıdaki gibi bir ekran çıktısı almak istiyoruz:

Gazi Üniversitesi
Gazi Üniversitesi
Gazi Üniversitesi
Gazi Üniversitesi
Gazi Üniversitesi
Gazi Üniversitesi
Gazi Üniversitesi
Gazi Üniversitesi
Gazi Üniversitesi
Gazi Üniversitesi
Gazi Üniversitesi

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

```
<?php
    for ($k=1;$k<=6;$k++)
    {
         print("<font size=$k>");
         echo "Gazi Üniversitesi";
         print("</font>");
         echo"<br>";
    }
    for ($k=5;$k>=1;$k--)
    {
         print("<font size=$k>");
         echo "Gazi Üniversitesi";
         print("</font>");
         echo"<br>";
    }
?>
```

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

Diğer bir örnek, bilinen faktöriyel işlemini gerçekleştirecek döngünün tanımlanmasıdır.

Örnekte, 8 sayısının faktöriyeli, yani 8X7X6X5X4X3X2X1 çarpımı hesaplanmaktadır.

```
<?php
    $sayi=8; $faktoryel=1;
    for ($i = 1; $i <= $sayi; $i++)
    {
        $faktoryel = $faktoryel * $i;
    }
    echo $faktoryel;
?>
```

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

Döngüleri iç içe kullanabiliriz. Bu durumda en içteki döngü, en hızlı dönen döngü olacaktır.

Bu mekanizmayı günümüzde de kullanılan elektrik ve su sayaçlarında bulunan mekanik sayaçları düşünebiliriz.

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

Üç basamaklı mekanik bir sayacın çalışma prensibi şu şekilde olacaktır: Birler hanesi,0 ile 9 arasındaki değerleri alırken, bu hanenin her turunda onlar hanesi 0 ile 9 arasındaki yeni değerini alacaktır.

Aynı durum onlar ve yüzler haneleri arasında gerçekleşecektir.

Bu çalışma biçiminin, içi içe üç For döngüsünden oluşan bir yapıda simülasyonunu gerçekleştirelim.

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

```
<?php
    for ($yuzler = 0; $yuzler <= 9; $yuzler++)
    {
         for ($onlar = 0; $onlar <= 9; $onlar++)
         {
              for ($birler = 0; $birler <= 9;
$birler++)
              {echo "$yuzler $onlar $birler <br>";
              }
         }
    }
?>
```

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

Biraz daha karmaşık bir örneği şu şekilde geliştirelim:

$$A^2+B^2=C^2$$

eşitliğini sağlayan pozitif tamsayılara Pisagor Üçlüleri adı verilir.

Örneğin $3^2+4^2=5^2$ gibi. Bu durumda (3,4,5) bir Pisagor üçlüsüdür.

Öyle bir kod yazalım ki, 1 ile 99 arasındaki bütün Pisagor üçlülerini bulsun ve sonuçları ekranda biçimlendirilmiş bir şekilde yazsın.

Akış Kontrol Deyimleri

```
<?php
     echo "<font size=\"5\">Pisagor Üçlüleri</font>";
     echo "<p>";
     $kareler="<font color=\"#000000\"><sup>2</sup></font>";
     $sayac=1;
     for ($a = 1; $a <= 99; $a++)
     {
            for ($b = 1; $b <= 99; $b++)
            {
                   for ($c = 1; $c <= 99; $c++)
                   {
                          if ($a*$a+$b*$b==$c*$c and $a<$b)
                          {
                                 echo $sayac++,"). <font
color=\"#FF0000\"><b>$a</b></font>",$kareler," + <font
color=\"#FF0000\"><b>$b</b></font>",$kareler," = <font
color=\"#FF0000\"><b>$c</b></font>",$kareler,"<br>";
                          }
                   }
            }
    }
?>
```

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

Yine sayılarla ilgili bir örnek verelim.

Kendisi hariç, 1 dahil olmak üzere, tam bölenlerinin toplamına eşit olan sayılara “mükemmel sayı” adı verilir. Örneğin 6 sayısının tam bölenleri 1,2 ve 3’dür.

6=1+2+3 olduğundan 6 bir mükemmel sayıdır.

Benzer şekilde 28 de bir mükemmel sayıdır.

Yazacağımız program, 1 ile 10.000 arasındaki mükemmel sayıları belirleyerek listeleyecektir.

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – For Döngüsü

```
<?php
    for ($sayi=6;$sayi<10000;$sayi++)
    {
          $toplam=1;
          for ($sayac=2;$sayac<=$sayi/2;$sayac++)
          {
               if ($sayi % $sayac == 0)
               { $toplam+=$sayac; }
          }
          if ($toplam==$sayi) {echo "$toplam <br>";}
    }
?>
```

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – Foreach Döngüsü

Genel yapısı aşağıdaki gibidir:

```
Foreach ($dizi as $deger)
 {
        Döngü tarafından çalıştırılacak kod;
 }
```

Giriş parametresi olarak dizi kullanan Foreach döngüsü, her bir tekrarda dizinin sıradaki değerini $deger değişkenine aktarır.

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – Foreach Döngüsü

```
<?php
    $dizi=array("bir","iki","üç","dört");
    foreach ($dizi as $deger)
    {
        echo "$deger<br>";
    }
?>
```

```
bir
iki
üç
dört
```

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – Foreach Döngüsü

```
<?php
    $a = array();
    $a[0][0] = "a";
    $a[0][1] = "b";
    $a[1][0] = "c";
    $a[1][1] = "d";
    foreach ($a as $v1) {
    foreach ($v1 as $v2) { echo "$v2<br>";
        }
    }
?>
```

```
a
b
c
d
```

Akış Kontrol Deyimleri

Döngüler – Sayaçlı Döngüler – Foreach Döngüsü

Burada $v1 değişkeni $a dizinin birinci ve ikinci boyutunu oluşturan iki diziyi temsil etmektedir. $v2 değişkeni ise bu tek boyutlu dizilerin terimlerini gösterir.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler

Koşullu döngülerde, tekrarlanacak kodun tekrar sayısı bir koşul tarafından belirlenir. Bu tür döngü yapılarında otomatik olarak artan bir sayaç değişkeni bulunmamaktadır.

Koşullu döngülerin iki alt türü vardır. Bunlardan ilki koşulun döngü başlangıcında denetlendiği yapılardır. Bu durumda koşul sağlanmadığı takdirde döngü içerisindeki kod hiçbir şekilde çalışmaz.

Diğer alt türde ise koşul, döngünün sonunda denetlenir. Birinci yapıdan farklı olarak bu durumda, koşul ne olursa olsun döngü içersindeki koşul en az bir defa çalıştırılacaktır.

Söz edilen döngü türlerinin PHP'deki karşılıkları sırasıyla While ve Do..While döngü ifadeleridir.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

Genel yapısı aşağıdaki gibidir:

```
while (koşul)
 {
        Döngü tarafından çalıştırılacak kod;
 }
```

Kelime anlamı “iken” olan while deyimi, döngünün etki alanında kalan kodu koşul sağlandığı sürece çalıştıracaktır. Bu yapıda otomatik olarak artan bir sayaç değeri bulunmamaktadır, eğer bir sayaç kullanılması gerekiyorsa bunun kontrolünün programcı tarafından ayrıca yapılması gerekmektedir.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

```
<?php
    $i=1;
    while ($i<=10) {
          echo "$i ";
          $i++;
    }
?>
```

```
1 2 3 4 5 6 7 8 9 10
```

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

Döngünün çalışma şeklini daha karmaşık bir örnek üzerinde inceleyelim:

153 sayısının aşağıdaki gibi bir aritmetik özelliği vardır:

$1^3 + 5^3 + 3^3 = 153$

Yani 153 sayısının basamaklarının küplerinin toplamı, yine sayının kendisine eşittir. Acaba 100 ile 999 arasında bu özelliğe sahip başka bir üç basamaklı sayı var mıdır?

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

Problemin çözümü için şu yolu izleyeceğiz:

Yüzler basamağını 1 ile 9, onlar ve birler basamaklarını ise 0 ile 9 arasında bir döngüye sokarak, iç-içe oluşturulan bu döngüsel yapı içerisinde basamakların küplerinin toplamının sayının bütününe eşit olup olmadığına bakacağız.

Eğer eşit ise bu sayının basamaklarını yazdıracağız. Sonuç olarak ekranda görüntülenen sayılar, 100 ile 999 arasında -eğer varsa- 153 ile aynı özelliğe sahip sayıların basamakları olacaktır.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

```
<?php
$yuzler = 1;
      while ($yuzler <= 9)
      {
              $onlar = 0;
              while ($onlar <= 9)
              {
                      $birler = 0;
                      while ($birler <= 9)
                      {
                              $sayi=$yuzler*100+$onlar*10+$birler;
                              if ($birler*$birler*$birler +
                              $onlar*$onlar*$onlar +
                              $yuzler*$yuzler*$yuzler==$sayi)
                              {
                                      echo "$yuzler - $onlar - $birler
                                        <br>";
                              }
                              $birler++;
                      }
                      $onlar++;
              }
              $yuzler++;
    }
?>
```

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

Kod çalıştırıldığında ekranda aşağıdaki sayılar görüntülenecektir:

```
1 - 5 - 3
3 - 7 - 0
3 - 7 - 1
4 - 0 - 7
```

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

Şimdiki örneğimiz, Hintli matematikçi Kaprekar (1905-1986) tarafından tanımlanan, dört basamaklı sayılara en fazla yedi kez aşağıdaki işlemler uygulandığında ortaya çıkan sabit 6174 sayısı üzerine olacaktır.

İşlemler , dört basamaklı sayılara uygulandığında en fazla yedi adımda sıfır veya 6174 sabit sayısını verir.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

1. Şu şartlara uygun dört basamaklı bir sayı alınır:

Tüm basamakları aynı sayıdan oluşmayan (2222 gibi - ilk işlem adımında sıfır sonucunu verecektir) veya herhangi üç basamağındaki sayılar aynı olup kalan bir basamaktaki sayı bu sayılardan bir büyük ya da bir küçük olmayan bir sayı (1112, 5565 veya 8788 gibi - ilk adımda 999 sayısını verecektir.)

2. Sayının basamaklarını büyükten küçüğe ve küçükten büyüğe doğru sıralayarak iki adet dört basamaklı sayı elde edilir.

3. Elden edilen sayılardan büyükten küçüğü çıkarılır.

4. İkinci adım tekrar edilir.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

Örneğin 3254 sayısı ile başlayalım:

1) 5432 - 2345 = 3087
2) 8730 - 378 = 8352
3) 8532 - 2358 = 6174

Görüldüğü gibi sadece üç adımda 6174 sabiti elde edilmiştir. Bu sayı için yönteme devam edilirse:

4) 7641 – 1467 = 6174

kısır döngüsüne ulaşılır. En fazla yedi adımda sıfır ya da 6174 sabit sayısı elde edilecek ve kısır döngüye girilecektir.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – While Döngüsü

```
<?php
      $sayi=1234;
      $adim=1;
      $dizi = array($sayi[0],$sayi[1],$sayi[2],$sayi[3]);
      while ($sayi <> 6174)
      {
            sort($dizi);
            $kucuk_sayi =
1000*(int)$dizi[0]+100*(int)$dizi[1]+10*(int)$dizi[2]+(int)$dizi[3];
            $buyuk_sayi =
1000*(int)$dizi[3]+100*(int)$dizi[2]+10*(int)$dizi[1]+(int)$dizi[0];
            $sayi=(string)($buyuk_sayi-$kucuk_sayi);
            echo "<p>$adim ) $buyuk_sayi - $kucuk_sayi = $sayi";
            $adim++;
            $dizi = array($sayi[0],$sayi[1],$sayi[2],$sayi[3]); }
?>
```

```
1) 4321 - 1234 = 3087
2) 8730 - 378 = 8352
3) 8532 - 2358 = 6174
```

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – Do..While Döngüsü

Genel yapısı aşağıdaki gibidir:

```
do      {
        Döngü tarafından çalıştırılacak kod;
        }
while (koşul);
```

Do..while döngüsünün While döngüsü ile arasındaki tek fark, koşulun döngü sonunda denetlenmesidir. Bunun sonucu olarak, bu yapıda döngünün etki alanında yer alan kod, koşulun sağlanıp sağlanmadığına bakmaksızın en az bir defa çalıştırılır.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – Do..While Döngüsü

While ve do..while döngü yapıları arasındaki farkı daha açık olarak gösterebileceğimiz bir örnek geliştirelim.

$a herhangi bir pozitif tamsayı olmak üzere, bu değerden başlayarak sıradaki her terimi aşağıdaki kurala elde edilecek bir sayı dizisi oluşturalım:

Eğer $a çift sayı ise bir sonraki terim=$a/2;

Aksi halde bir sonraki terim=3*$a+1.

Örneğin $a=5 olsun. Bu sayı tek olduğundan bir sonraki terim, 3*5+1=16 olacaktır.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – Do..While Döngüsü

16 çift olduğundan bir sonraki terim 16/2=8 olacaktır. Böylece dizinin terimleri,

5,16,8,4,2,1,4,2,1,4,2,1,…

biçiminde gidecektir. Görüldüğü gibi dizinin terimi 1'e ulaştıktan sonra 4,2,1 kısır döngüsüne ulaşmaktadır. Üstelik hangi değerle başlanırsa başlansın, sonuçta dizi mutlaka bu kısır döngüye takılıp kalmaktadır.

Bilgisayarlar kullanılarak milyonlarca başlangıç değeri için dizinin 1 değerine ulaştığının gözlemlenmesine karşın, bunun her pozitif tamsayı için böyle olacağı günümüze kadar matematiksel olarak kanıtlanamamıştır. «Collatz Problemi» olarak bilinen bu problem, henüz çözülmemiş ödüllü bir problemdir.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – Do..While Döngüsü

Aşağıda aynı örneğin iki döngü yapısı kullanılarak geliştirilmiş biçimleri yer almaktadır.

```
<?php
	$a=10;
	while ($a!=1)
	{
		if ($a % 2 == 0) {$a=$a/2;}
		else {$a=3*$a+1;}
		echo "$a ";
	}
?>
```

```
<?php
	$a=10;
	do
	{
		if ($a % 2 == 0) {$a=$a/2;}
		else {$a=3*$a+1;}
		echo "$a ";
	}
	while ($a!=1)
?>
```

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – Do..While Döngüsü

Bu şekliyle her iki kod çalıştırıldığında da aynı sonuç elde edilir:

```
5 16 8 4 2 1
```

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – Do..While Döngüsü

Ancak örnek kodlarda $a başlangıç değeri,

$a=1;

olarak belirlendiğinde iki döngü yapısının çalışma sistemleri arasındaki fark ortaya çıkacaktır.

Bu durumda while döngüsünün kullanıldığı 02-047 numaralı örnek kod herhangi bir çıktı üretmezken, do..while döngüsünün kullanıldığı 02-048 numaralı örnek kodun çıktısı,

```
4 2 1
```

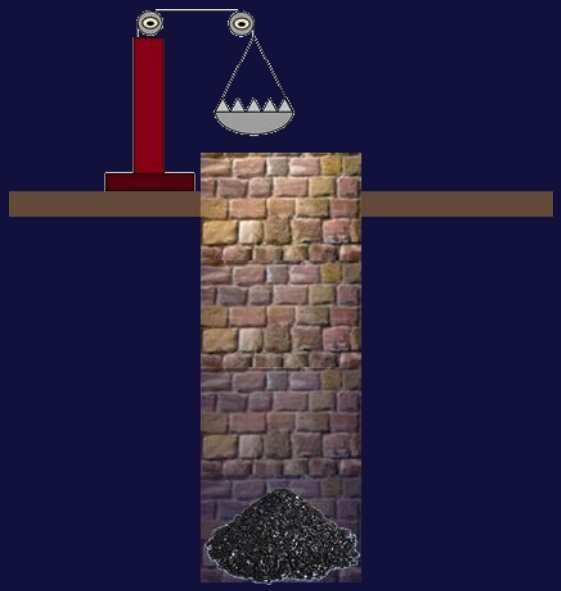
olur.

Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – Do..While Döngüsü

Bu örneğimizde bir kömür kuyusu ve o kuyudan yukarıya kömür çıkaran bir kepçe bulunmaktadır.

Kepçe, kuyunun dibinden almış olduğu kömürü yukarı çıkarırken her metrede yarısını dökmektedir.



Akış Kontrol Deyimleri

Döngüler – Koşullu Döngüler – Do..While Döngüsü

Bu durumda kepçenin almış olduğu kömür miktarı ve kuyu derinliği bilindiğinde yukarıya çıkacak kömür miktarını hesaplayacak PHP kodunu yazalım.

```
<?php
    $d=6;     // Kuyunun derinliği.
    $m=1100; // Kepçeye alınan kömür miktarı.
    do
    {
          $m/=2;
          $d--;
    }
    while ($d!=0);
   echo "Yukarı çıkan kömür miktarı = $m";
?>
```

Örnek Uygulama: Sihirli Kare

Şu ana kadar edindiğimiz bilgileri kullanabileceğimiz kapsamlı bir örnek uygulama geliştirelim.

Amacımız, sihirli kare adı verilen ve sayılardan oluşan karelerin bilgisayar tarafından oluşturulmasını sağlayacak bir program yazmak. Bu sayı karelerinin özelliği, satır, sütun ve köşegen üzerindeki sayıların toplamlarının eşit olmasıdır.

Sihirli karelerin inşası için bir çok algoritma bulunmaktadır. Bizim kullanacağımız algoritma, n bir tek sayı olmak üzere, nxn boyutunda bir sihirli karenin oluşturulmasını sağlamaktadır.

Örnek Uygulama: Sihirli Kare

Algoritma, 1'den başlayarak bütün kare doluncaya kadar sayıların kare üzerine belirli kurallar dahilinde yerleştirilmesi ilkesine dayanır. Bu kuralları aşağıdaki gibi sıralayabiliriz:

KURALLAR

1. İlk sayı olan 1, karenin ilk satırının orta hücresine yerleştirilir.
2. Sıradaki her sayı, kendisinden önceki sayının sağ-üst köşesinde yer alan hücreye yerleştirilir.

Bu kuralların işletilmesi sırasında ortaya çıkabilecek dört aykırı durum vardır:

Örnek Uygulama: Sihirli Kare

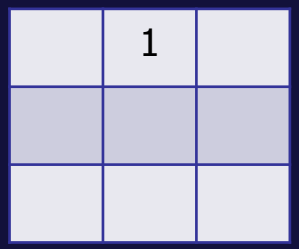
AYKIRI DURUMLAR

1. Sıradaki sayıyı yerleştireceğimiz hücre dolu olabilir. Bu durumda sıradaki sayı, bir önceki sayının bulunduğu hücrenin altındaki hücreye yerleştirilir.
2. Sıradaki sayıyı yerleştireceğimiz hücre karenin üstünden dışında kalıyor olabilir. Bu durumda sıradaki sayı, dışarıda kalan hücrenin hizasındaki sütunun en altındaki hücreye yerleştirilir.
3. Sıradaki sayıyı yerleştireceğimiz hücre karenin sağ tarafından dışında kalıyor olabilir. Bu durumda sıradaki sayı, dışarıda kalan hücrenin hizasındaki satırın en solundaki hücreye yerleştirilir.
4. Sıradaki sayıyı yerleştireceğimiz hücre karenin sağ-üst çaprazından dışında kalıyor olabilir. Bu durumda sıradaki sayı, bir önceki sayının bulunduğu hücrenin altındaki hücreye yerleştirilir.

Örnek Uygulama: Sihirli Kare

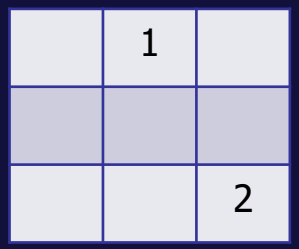
Üç satır ve üç sütundan oluşan sihirli kareyi algoritmanın kurallarını uygulayarak oluşturalım.

Öncelikle ilk sayımız olan 1'i karenin ilk satırının orta hücresine yerleştirelim:



Örnek Uygulama: Sihirli Kare

Sıradaki sayı olan 2, 1 sayısının bulunduğu hücrenin sağ-üst köşesindeki hücreye yerleşecek. Ancak burada, karenin üzerinden dışarıya çıkıldığı için 2 sayısı dışarıya çıkılan hizadaki sütunun en alt hücresine yerleştiriliyor:



Örnek Uygulama: Sihirli Kare

Sıradaki sayı olan 3, 2 sayısının bulunduğu hücrenin sağ-üst köşesindeki hücreye yerleşecek. Ancak burada, karenin sağ tarafından dışarıya çıkıldığı için 3 sayısı dışarıya çıkılan hizadaki satırın en solundaki hücreye yerleştiriliyor:

	1	
3		
		2

Örnek Uygulama: Sihirli Kare

Sıradaki sayı olan 4, 3 sayısının bulunduğu hücrenin sağ-üst köşesindeki hücreye yerleşecek. Ancak bu hücre dolu olduğu için 4 sayısı 3 sayısının bulunduğu hücrenin altında yer alan hücreye yerleştiriliyor:

	1	
3		
4		2

Örnek Uygulama: Sihirli Kare

Sıradaki sayı olan 5, 4 sayısının bulunduğu hücrenin sağ-üst köşesindeki hücreye yerleştiriliyor:

	1	
3	5	
4		2

Örnek Uygulama: Sihirli Kare

Sıradaki sayı olan 6, 5 sayısının bulunduğu hücrenin sağ-üst köşesindeki hücreye yerleştiriliyor:

	1	6
3	5	
4		2

Örnek Uygulama: Sihirli Kare

Sıradaki sayı olan 7, 6 sayısının bulunduğu hücrenin sağ-üst köşesindeki hücreye yerleşecek. Ancak burada, karenin sağ-üst çaprazından dışarıya çıkıldığı için 7 sayısı 6 sayısının bulunduğu hücrenin altında yer alan hücreye yerleştiriliyor:

	1	6
3	5	7
4		2

Örnek Uygulama: Sihirli Kare

Sıradaki sayı olan 8, 7 sayısının bulunduğu hücrenin sağ-üst köşesindeki hücreye yerleşecek. Ancak burada, karenin sağ tarafından dışarıya çıkıldığı için 8 sayısı dışarıya çıkılan hizadaki satırın en solundaki hücreye yerleştiriliyor:

8	1	6
3	5	7
4		2

Örnek Uygulama: Sihirli Kare

Ve son olarak sıradaki sayı olan 9, 8 sayısının bulunduğu hücrenin sağ-üst köşesindeki hücreye yerleşecek. Ancak burada, karenin üzerinden dışarıya çıkıldığı için 9 sayısı dışarıya çıkılan hizadaki sütunun en alt hücresine yerleştiriliyor:

8	1	6
3	5	7
4	9	2

Örnek Uygulama: Sihirli Kare

Sonuç olarak elde ettiğimiz sihirli karede her satırda, her sütunda ve her iki köşegende yer alan hücrelerdeki sayıların toplamı aynıdır: **15**

Şimdi bu algoritmanın PHP'de kodlamasını gerçekleştireceğiz. Programımız, bizim için n=11 boyutlu sihirli kareyi oluşturacaktır.

Programda ağırlıklı olarak akış kontrol deyimlerini ve dizileri kullanacağız.

Örnek Uygulama: Sihirli Kare

```
<?php
	$dizi1=range(1,11);
	$dizi2=range(1,11);
	$dizi=array($dizi1,$dizi2);
	$sayi=11;
	$sayikare=$sayi*$sayi;
	$toplam=0;
	for($i=1;$i<=$sayi;$i++)
	{
		for($j=1;$j<=$sayi;$j++)
		{
			$dizi[$i][$j]=0;
		}
	}
	$i=1;
	$j=floor(($sayi+1)/2);

	for ($k=1;$k<=$sayikare;$k++)
	{
		if ($dizi[$i][$j]>0)
		{
			$i=$i+2;
			$j=$j-1;
		}
		$dizi[$i][$j]=$k;
		$i=$i-1;
		$j=$j+1;
		...
```

Örnek Uygulama: Sihirli Kare

```
...
                if (($i==0) and ($j>$sayi))
                {
                        $i=$i+2;
                        $j=$j-1;
                }
                if ($i==0)        {$i=$sayi;}
                if ($j>$sayi) {$j=1;}
                $toplam=$toplam+$k;
        }
        echo "<table border=\"2\" align=\"center\"><tr>";
        for($i=1;$i<=$sayi;$i++)
        {
                echo "</tr><tr>";
                for($j=1;$j<=$sayi;$j++)
                {
                        $a=$dizi[$i][$j];
                        print "<td align=\"center\">$a</td>";
                }
        }

        $stoplam=$toplam/$sayi;
        echo "<p>Toplamlar : $stoplam";
        echo "<p>";
        echo "</table>";
?>
```

Örnek Uygulama: Sihirli Kare

Örnek kodda iki boyutlu dizi tanımlaması alternatif bir yolla yapılmıştır:

```
$dizi1=range(1,11);
$dizi2=range(1,11);
$dizi=array($dizi1,$dizi2);
```

Bu yöntemde öncelikle range fonksiyonu kullanılarak uzunluğu 11 terimden oluşan iki tane tek-boyutlu dizi tanımlanmakta, daha sonra bunlar kullanılarak 11×11=121 terimden oluşan iki boyutlu dizimiz tanımlanmaktadır.

Fonksiyonlar

Dinamik olarak kullanılabilen altprogramlar biçiminde tanımlayabileceğimiz fonksiyonlar, PHP dilinin gücünü aldığı en önemli bileşenlerdir. Şöyle ki, günümüzde yazılmış 700'den fazla dahili PHP fonksiyonu bulunmaktadır. Bunlara anonim olarak yazılmış ve İnternetten ulaşılabilen binlerce kullanıcı-tanımlı fonksiyonu da katarsak, konunun önemi ortaya çıkacaktır.

Fonksiyonları yazma amacımız, çeşitli programlar tarafından kullanılan, ya da aynı program içerisinde tekrarlanması gereken işlevsel kodları tekrar kullanılabilir programcıklar olarak oluşturmaktır. Bu programcıklar, ana program ile iki yönlü olarak veri alışverişinde bulunabilirler. Kendi kendilerine çalışmazlar, sadece ana program tarafından çağırıldıklarında çalışırlar.

Fonksiyonlar

Kullanıcı-tanımlı fonksiyonlar, PHP dilini kullanarak uygulama geliştirenler tarafından yazılan fonksiyonlardır. Bunlar İnternet üzerinden yoğun olarak paylaşılmaktadır.

Dahili fonksiyonlar ise doğrudan PHP çözümleyicisi tarafından denetlenen, PHP geliştirme grubu tarafından yazılmış kullanıma hazır fonksiyonlardır.

Fonksiyonlar

Kullanıcı-Tanımlı Fonksiyonlar

Kendi programlarınızda kullanmak üzere, kendi fonksiyonlarınızı yazabilirsiniz. Bunun için genel olarak aşağıdaki gibi bir tanımlama yapmanız yeterli olacaktır:

```
function fonksiyonAdı()
{
        fonksiyon tarafından çalıştırılacak kodlar;
}
```

Bu şekilde tanımlanan bir fonksiyonu çağırmak için adını yazmamız yeterli olacaktır.

Basit bir örnekle başlayalım. İşlevi ekrana adınızı yazmak olan bir fonksiyon tanımlayalım ve onu ana programımızdan çağırarak kullanalım.

Fonksiyonlar

```
<html>
<body>

<?php
function adimiYaz()
{
    echo "Murat Sönmez";
}
    echo "Benim adım ";
    adimiYaz();
?>

</body>
</html>
```

```
Benim adım Murat Sönmez
```

Fonksiyonlar

Kullanıcı-Tanımlı Fonksiyonlar

Kırmızı renkle gösterilen fonksiyon tanımı, PHP tarafından doğrudan çalıştırılmaz. Ana program bloğu içersinden çağırıldığında çalıştırılacaktır.

Fonksiyonlar

Kullanıcı-Tanımlı Fonksiyonlar – Parametreler ve Dönen Değerler

Fonksiyonlar, ana programdan farklı türlerde veriyi alabilirler. Değerleri ana programdan fonksiyonlara taşıyan bu yapılara parametre adı verilir.

Bir fonksiyon, parametreleri kullanarak ya doğrudan bir işlevi yerine getirir, ya da ana program için bir değer üretir. Eğer bir değer üretmiş ise bu değerin ana programa döndürülmesi için kullanılan deyim, «return» deyimidir.

Fonksiyonlar

Kullanıcı-Tanımlı Fonksiyonlar – Parametreler ve Dönen Değerler

```
<html> <body>
<?php
function adimiYaz($ad)
{
    echo "$ad Sönmez";
}
    echo "Benim adım ";  adimiYaz("Murat");
    echo "<br>Kardeşimin adı ise ";
    adimiYaz("Merve");
?>
</body> </html>
```

```
Benim adım Murat Sönmez
Kardeşimin adı ise Merve Sönmez
```

Fonksiyonlar

Kullanıcı-Tanımlı Fonksiyonlar – Parametreler ve Dönen Değerler

Aşağıdaki örnekte ise, fonksiyon tarafından üretilen değer ana programa gönderilmektedir.

```
<?php
    $Gelir=1100;
    function vergiyi_dus ($Geliriniz)
    {     $Geliriniz = $Geliriniz -
                          (($Geliriniz/100)*20);
          return $Geliriniz; }
    echo "$Gelir TL gelirin vergi düşülmüş hali ";
    echo(vergiyi_dus ($Gelir)); echo " TL dir.";
?>
```

```
1100 TL gelirin vergi düşülmüş hali 880 TL dir.
```

Fonksiyonlar

Kullanıcı-Tanımlı Fonksiyonlar – Parametreler ve Dönen Değerler

Diğer bir örnek olarak, daha önce yapmış olduğumuz tek basamaklı bir sayının Türkçe okunuşunu yazacak programı, bu defa fonksiyon kullanarak tasarlayalım.

```
<?php

function turkce_oku ($Gelen_Sayi){
switch ($Gelen_Sayi) {
case 1:
    $bir="BİR";
    break;
case 2:
    $bir="İKİ";
    break;
```

Fonksiyonlar

Kullanıcı-Tanımlı Fonksiyonlar – Parametreler ve Dönen Değerler

```
case 3:
    $bir="ÜÇ";
    break;
case 4:
    $bir="DÖRT";
    break;
case 5:
    $bir="BEŞ";
    break;
case 6:
    $bir="ALTI";
    break;
case 7:
    $bir="YEDİ";
    break;
```

Fonksiyonlar

Kullanıcı-Tanımlı Fonksiyonlar – Parametreler ve Dönen Değerler

```
case 8:
    $bir="SEKİZ";
    break;
case 9:
    $bir="DOKUZ";
    break; }
return $bir;}

$Sayi=8;
echo "$Sayi Sayısının Türkçe Okunuşu : "; echo(turkce_oku
($Sayi));

?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Dahili fonksiyonlar, sayıları itibarı ile herhangi bir ders kapsamında bütün detayı ile ele alıp anlatılabilecek bir konu değildir. Bu fonksiyonların öğrenilmesi daha çok, gereksinim duydukça referans kaynakları vasıtasıyla gerçekleşmektedir.

Bu bölümde, dahili fonksiyonların kullanımına yönelik bazı örnek programlar verilmiştir.

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Aşağıdaki örnek program öncelikle scandir klasör fonksiyonunu kullanarak etkin klasörde yer alan “ödevler” klasörünün içeriğini almakta, daha sonra print_r dizi fonksiyonu bu bilgiyi bir dizi formatında ekrana yazdırmaktadır.

```
<?php
    print_r(scandir("odevler"));
?>
```

```
Array
(
[0] => .
[1] => ..
[2] => 050535002.rar
[3] => 050535003.rar
[4] => 050535005.rar
[5] => 050535006.rar
[6] => 050535007.rar
[7] => 050535010.rar
)
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Aşağıdaki örnekte programın çalışması bir hata (error) fonksiyonu kullanılarak durdurulmakta ve hata mesajı verilmektedir.

```
<?php
    $sayi=1;
    if ($sayi<2){
        trigger_error("Kullanıcı-tanımlı hata
durumu tetiklendi.");}
?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Yuvarlama fonksiyonlarının kullanıldığı bir örnek aşağıda verilmiştir.

```
<?php
  echo "ceil(6.1)=",ceil(6.1); echo "<hr>";
  echo "ceil(7.9)=",ceil(7.9); echo "<hr>";
  echo "round( 13.4)=",round( 13.4); echo "<hr>";
  echo "round( 32.5)=",round( 32.5); echo "<hr>";
  echo "floor(8.1)=",floor(8.1); echo "<hr>";
  echo "floor(4.9)=",floor(4.9); echo "<hr>";
?>
```

```
ceil(6.1)=7
---------------------------------------------
ceil(7.9)=8
---------------------------------------------
round( 13.4)=13
---------------------------------------------
round( 32.5)=33
---------------------------------------------
floor(8.1)=8
---------------------------------------------
floor(4.9)=4
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Aşağıdaki örnek uygulama, trigonometrik fonksiyonları kullanarak ekranda bir trigonometrik değerler tablosu oluşturmaktadır.

```
<?php
/* 0-360 derece arasındaki bütün açıların sinüs, kosinüs ve
tanjant değerleri hesaplanıyor. */
echo "<h1>Trigonometrik Deðerler Tablosu</h1>
<table border=1>
    <tr bgcolor=#AEEEF9>
      <td width=70>Açý</td>
      <td width=110>Sinüs</td>
      <td width=110>Kosinüs</td>
      <td width=110>Tanjant</td>
    </tr>";
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
      for ($aci=0;$aci<=360;$aci++)
      {
            $sinus = number_format(sin(deg2rad($aci)),6);
            $kosinus = number_format(cos(deg2rad($aci)),6);
            if (!($aci==90) and !($aci==270))
            {

      $tanjant=number_format(tan(deg2rad($aci)),6);
            }
            else{ $tanjant="=sonsuz=";}
            echo "
                        <tr bgcolor=#CACBEE>
                        <td width=70>$aci</td>
                        <td width=110>$sinus</td>
                        <td width=110>$kosinus</td>
                        <td width=110>$tanjant</td>
                  </tr> "; }
?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Örneğin ekran çıktısı aşağıdaki gibi olacaktır.

```
  Trigonometrik Değerler Tablosu
Açı       Sinüs     Kosinüs   Tanjant
0         0.000000  1.000000  0.000000
1         0.017452  0.999848  0.017455
2         0.034899  0.999391  0.034921
3         0.052336  0.998630  0.052408
4         0.069756  0.997564  0.069927
5         0.087156  0.996195  0.087489
6         0.104528  0.994522  0.105104

...       ...       ...       ...
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Daha önce tek basamaklı sayılar için yaptığımız, bir sayının Türkçe okunuşu örneğini, bazı dahili fonksiyonları kullanarak iki basamaklı sayılar için yapalım.

```
<?php
function turkce_oku ($Gelen_Sayi){
$birler = fmod($Gelen_Sayi,10);
$onlar = floor($Gelen_Sayi/10);
switch ($onlar) {
case 1:
    $on="ON";
    break;
case 2:
    $on="YİRMİ";
    break;
case 3:
    $on="OTUZ";
    break;
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
case 4:
    $on="KIRK";
    break;
case 5:
    $on="ELLİ";
    break;
case 6:
    $on="ALTMIŞ";
    break;
case 7:
    $on="YETMİŞ";
    break;
case 8:
    $on="SEKSEN";
    break;
case 9:
    $on="DOKSAN";
    break; }
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
switch ($birler) {
case 1:
    $bir="BİR";
    break;
case 2:
    $bir="İKİ";
    break;
case 3:
    $bir="ÜÇ";
    break;
case 4:
    $bir="DÖRT";
    break;
case 5:
    $bir="BEŞ";
    break;
case 6:
    $bir="ALTI";
    break;
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
case 7:
    $bir="YEDİ";
    break;
case 8:
    $bir="SEKİZ";
    break;
case 9:
    $bir="DOKUZ";
    break; }
return $on.$bir; }
$Sayi=48;
echo "$Sayi Sayısının Türkçe Okunuşu : "; echo(turkce_oku
($Sayi));
?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Bu örnekte öncelikle iki basamaklı sayı onlar ve birler basamaklarına ayrılıyor, daha sonra onlar ve birler basamağına karşılık gelen Türkçe okunuşlar belirlenerek birleştiriliyor.

Burada dikkat edilmesi gereken nokta, iki basamaklı sayının nasıl basamaklarına ayrıldığıdır.

Sayının 10 modundaki değeri, doğrudan birler basamağını verir:

43 mod 10 = 3 gibi…

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Bu işlem PHP'de fmod dahili fonksiyonu kullanılarak gerçekleştiriliyor:

$birler = fmod($Gelen_Sayi,10);

Onlar basamağını ise sayıyı 10'a bölüp, çıkan sonucun tam kısmını alarak elde ediyoruz.

43 / 10 = 4.3

Tam kısım (4.3) = 4 gibi…

Bu işlem PHP'de floor dahili fonksiyonu kullanılarak gerçekleştiriliyor:

$onlar = floor($Gelen_Sayi/10);

Böylece iki basamaklı sayımızı basamaklarına ayırmış olduk.

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Karekökü tamsayı olan sayılara “tamkare sayılar” adı verilir. Örneğin 4, karekökü olan 2 tamsayı olduğundan bir tamkaredir. 9, 16 ve 25 de tamkare sayılardır, ancak 5,6,7,8 değildir.

Şimdi yazacağımız PHP programı, 1 ile 1000 arasındaki tamkare sayıları listeleyecektir.

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
<?php
     $sayi = 1;
     while ( $sayi <= 1000 )
     {
     if (floor(sqrt($sayi))==sqrt($sayi))
     {     echo "$sayi", "<br>";   }
     $sayi ++;
     }
?>
```

Bu örnekte dikkat etmemiz gereken nokta, sayının karekökünün tamsayı olup olmadığının anlaşılması için PHP fonksiyonlarının nasıl kullanıldığıdır.

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Bu işlem için iki matematik fonksiyonu kullanılmıştır: floor ve sqrt.

Eğer bir sayının karekökünün tam kısmı, sayının karekökünü eşit ise o sayı bir tamkare sayıdır:

```
if (floor(sqrt($sayi))==sqrt($sayi))
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Şimdi string fonksiyonlarını kullanan bazı örnek programlar tasarlayalım.

İlk örneğimiz, 1949 yılında Hintli matematikçi Kaprekar tarafından tanımlanan Kaprekar sayıları ile ilgili olsun.

n basamaklı bir t Kaprekar sayısının karesi alınıp sağdaki n basamağı solda kalan n-1 basamağa eklendiğinde sonuç yine t sayısını verir.

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Örneğin; 55, iki basamaklı bir sayıdır.

55^2 = 3025, sağdan iki basamak 25, soldan iki basamak 30.

Bu iki sayının toplamı 30+25=55 yani sayının kendisidir.

1, 9, 45, 55, 99, 297, 703, 999, 2223, 2728, 4950 sayıları da diğer bazı Kaprekar sayılarıdır.

Yazacağımız program, bir for döngüsü içerisinde 1 ile 100.000 arasındaki bütün Kaprekar sayılarını substr fonksiyonunu kullanarak listelesin.

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
<?php
  $ust = 100000;
  for ($sayi = 9; $sayi <= $ust; $sayi++)
  {
      $basamak = strlen($sayi);
      $sayi2 = $sayi * $sayi;
      $basamak2 = strlen($sayi2);
      $soltaraf = substr($sayi2, 0, $basamak2-
$basamak);
      $sagtaraf = substr($sayi2, $basamak2-$basamak,
$basamak);
      $toplam = $soltaraf + $sagtaraf;
      if ($toplam == $sayi && $sayi != 10)
      {   echo "BULUNDU : $sayi <br>";   }
  }
?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Bu örneğimizde konu metni içerisinde SPAM denetimi yapılmaktadır.

```
<?php
    $konu="Bugünün şanslı insanı sizsiniz";
    if ((strpos ($konu,"kazandınız") != 0) ||
(strpos ($konu,"şanslı") != 0) ||  (strpos
($konu,"1000") != 0))
    { echo "Bu e-posta muhtemelen bir SPAM!";   }
    else { echo "Güvenli bir e-posta."; }
?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Bu örneğimizde, kullanıcının adını, soyadını ve e-posta uzantısını kullanarak e-posta adresini otomatik olarak oluşturacak programı yazalım.

Uygulamamız e-posta adresini şu kurala göre oluştursun:

<Adın ilk harfi>.<Soyad>@<e-posta uzantısı>

Örneğin adı Çetin, soyadı Öztürk olan bir kullanıcı için, gazi.edu.tr e-posta uzantısına göre,

c.ozturk@gazi.edu.tr

e-posta adresi oluşturulsun.

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Burada çözmemiz gereken en önemli problem, Türkçe harflerin benzer İngilizce harflere dönüştürülmesidir. Örneğin “ğ” harfinin “g” harfine dönüştürülmesi gibi.

Bunu bizim için, yazacağımız bir dönüşüm fonksiyonu yapacaktır.

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
<?php
function turkceharf_donustur ($harf)
{    switch ($harf) {
     case "ç":
     $harf="c";
     break;
     case "ğ":
     $harf="g";
     break;
     case "ö":
     $harf="o";
     break;
     case "ş":
     $harf="s";
     break;
     case "ü":
     $harf="u";
     break;
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
case "İ":
     $harf="i";
     break;
    case "ı":
     $harf="i";
     break;
     }
     return $harf;}
function eposta_olustur ($adi, $soyadi)
{
     $adilk=strtolower(substr($adi,0,1));
     $soyad=strtolower($soyadi);
     $eposta = $adilk.".".$soyad;
     $yeni_eposta="";
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
     for ($i = 0; $i <= strlen($eposta); $i++)
     {
     $yeni_eposta=$yeni_eposta.turkceharf_donustur(substr(
$eposta,$i,1));
     }
     return $yeni_eposta;
}
$ad = "Çetin";
$soyad = "Ceviz";
$e_posta = "gazi";
echo "<p>e-posta adresi : "; echo
eposta_olustur($ad,$soyad)."@".$e_posta;
?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Yazdığımız turkceharf_donustur() fonksiyonu, kendisine gelen Türkçe harfi benzeri İngilizce harfe dönüştürmektedir.

Önce adın ilk harfi ile soyad küçük harfe dönüştürülerek ve araya nokta işareti konularak birleştirilmektedir. Daha sonra oluşturulan bu string, harf harf fonksiyonumuza gönderilmekte ve içerisinde bulunan Türkçe harfler dönüştürülmektedir.

Sonuç olarak elde edilen string, e-posta uzantısı ile birleştirilmek suretiyle e-posta adresi elde edilmektedir.

Aynı işleve sahip, ancak kodu önemli ölçüde azaltacak başka bir yöntemde, dönüştürme işleminde diziler kullanılmaktadır.

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
<?php
function Tr_Donustur($gelen_metin)
{    $gelen_metin = trim($gelen_metin);
     $aranan_harfler = array('ç','ğ','ı','ö','ş','ü','İ');
     $degistirilecek_harfler =
array('c','g','i','o','s','u','i');
     return
str_replace($aranan_harfler,$degistirilecek_harfler,$gelen_
metin);
}
if(isset($_POST["dugme"]))
  { $ad = $_POST['ad'];
     $soyad = $_POST['soyad'];
     $e_posta = $_POST['eposta'];
     echo "<p>e-posta adresi : "; echo
Tr_Donustur(strtolower($ad[0].".".$soyad))."@".$e_posta;
  }
?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

strstr() fonksiyonunun kullanıldığı başka bir örnek:

```
<?php
     $sifreniz = "tolgaguyer";
     $adiniz = "tolga";
     if ( strstr( $sifreniz, $adiniz ) )
     print "Şifreniz içersinde adınız yer
                 almamalıdır!";
     else
     print "Teşekkürler...";
?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Kullanıcı tarafından girilen bir metni Morse alfabesine göre kodlayan bir örnek:

```
<?php
    $morseKodlari = array ('A' => '.-', 'B' => '-...', 'C'
=> '-.-.', 'D' => '-..', 'E' => '.', 'F' => '..-.', 'G' =>
'--.', 'H' => '....', 'I' => '..', 'J' => '.---','K' => '-
.-', 'L' => '.-..', 'M' => '--', 'N' => '-.', 'O' => '---',
'P' => '.--.', 'Q' => '--.-', 'R' => '.-.', 'S' => '...',
'T' => '-', 'U' => '..-', 'V' => '...-', 'W' => '.--', 'X'
=> '-..-', 'Y' => '-.--', 'Z' => '--..', '0' => '-----',
'1' => '.----', '2' => '..---', '3' => '...--', '4' =>
'....-', '5' => '.....', '6' => '-....', '7' => '--...',
'8' => '---..', '9' => '----.', '.' => '.-.-.-', ',' => '--
..--', ':' => '---...', '?' => '..--..', '-' => '-....-',
'/' => '-..-.', '(' => '-.--.-', ' ' => '(bosluk)');
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

```
	$metin='Bu bir denemedir.';
	$uzunluk=strlen($metin);
	$morseKodu='';
	for($i=0;$i<=$uzunluk-1;$i++)
	{
		$morseKodu=$morseKodu .
$morseKodlari[strtoupper($metin[$i])] . ' ';
	}
	echo("<strong> $morseKodu </strong>");
?>
```

Fonksiyonlar

Dahili (İçsel) Fonksiyonlar

Bu örnekte dikkat etmemiz gereken önemli bir nokta,

($metin[$i])

kullanımdır. Burada $metin bir string değişkenidir ancak $i indisi ile bir dizi gibi kullanılmıştır.

Buradan çıkaracağımız sonuç, string türündeki her değişkenin, karakterlerden oluşmuş bir dizi olarak kullanılabileceği gerçeğidir.

Bu özellik, stringler üzerinde işlem yaparken burada olduğu gibi bize önemli kolaylıklar sağlamaktadır.