



## For ve Foreach Döngüleri

26.02.2007

for ve foreach döngülerinin kullanımını sırasıyla inceleyelim.

```
for ( i = 0 ; i < hedef ; i++)  
{  
    // i değişkeni ile yapılacak işlemler  
}
```

i döngü değişkenine genellikle başlangıçta 0 atanır. Dolayısıyla bir dizinin indekslenmesi ile ilgili işlemlerde kolaylık sağlar.

Çoğu programcı while-do döngüsü kullanmaktansa for döngüsü kullanmayı tercih eder. Hatta bazı yerlerde while-do döngüsünü kullanmak daha uygun olduğu halde for döngüsü kullanırlar.

For döngüsünün de while döngüsünde olduğu gibi karmaşık kullanım şekilleri mevcuttur. Örneğin break ve exit cümlecikleri while döngüsündeki görevinin aynısını for döngüsünde de üstlenmektedir.

For kelimesinden sonra parantez açılmaldır. Parantez içerisine noktalı virgüller ile ayrılmış üç ifade yazılır. Bu ifadeler sırasıyla başlangıç değeri, koşul ve değişim değeridir.

Parantez içindeki üç ifadeden ilki ile döngünün başlangıç değerini belirtiriz. Genellikle tek bir değişken kullanılır, fakat isterseniz aşağıdaki gibi birden fazla değişken için başlangıç değeri atayabilirsiniz.

```
for ( i = 0 , j = 17 ; ..... ; .... )
```

Parantez içindeki ikinci ifade ile koşul durumunu belirtiriz. Bu bölüm boolean türünde kontrol edilir. Şart sağlanmış ise false değeri dönecek ve döngüden çıkmış olacaktır. Eğer koşul henüz gerçekleşmemiş ise true değerini alır ve döngü işlemeye devam eder.

```
for ( i = 0 ; false ; i++)  
{  
    // işleme alınmayacaktır.  
}
```

Parantez içindeki üçüncü ifade ile değişim miktarını belirtiriz. Başlangıç değerinde olduğu gibi birden fazla değişkenin değişimi eklenebilir.

```
for ( i = 0 , j = 100 ; i < j ; i++ , j -= 5 )  
{  
    // i ve j değişkenleri kullanılabilir  
}
```

Bu üç ifade belirtilirken kullandığımız değişkenlerin başlangıç değerlerini daha önceden atayabiliriz. Aşağıdaki örnek bir önceki ile aynı sonucu verecektir.

```
j = 100 ;  
  
for ( i = 0 ; i < j ; i++)  
{  
    // i ve j değişkenleri kullanılabilir  
  
    j -= 5 ;  
}
```

```
}
```

İsterseniz bu üç ifade yerini boş bırakıp sadece noktalı virgülleri parantezle içerisinde bırakabilirsiniz.

```
for (;;)
{
    // herhangi bir işlem..
}
```

Bu şekilde sonsuz döngü oluşturabilirsiniz. While döngüsü ile ise sonsuz döngü aşağıdaki şekilde olacaktır.

```
while (true)
{
    // Herhangi bir işlem.
}
```

Başlangıç değerini belirtirken kullandığımız i değişkeni program içerisinde daha önceden tanımlamadıysanız döngü içerisinde aşağıdaki şekilde belirtmelisiniz.

```
for (i = 0; i < hedef; i++)
{
    // i değişkeni ile yapılan işlemler
}
```

Örneğin string bir dizi olan sdizi için döngü aşağıdaki şekilde olacaktır.

```
for (int i = 0; i < sdizi.Length; i++)
{
    Console.WriteLine(sdizi[i]);
}
```

Bu döngüde tek bir ifade olduğundan süslü parantezleri kullanmayabilirsiniz.

Yukarıdaki örneklerde i değişkeninin değeri, eğer döngü herhangi bir sebeple yarıda sonlandırılmadıysa, döngü sonucunda hedef değerine eşit olur.

Eğer C veya C++ dillerini kullanan deneyimli bir programcıya 1 den 100 'e kadar olan sayıları C# kodu ile ekrana yazdırmasını isterseniz, aşağıdaki şekilde bir sonuç alırsınız:

```
for (int i = 0; i < 100; i++)
    Console.WriteLine(i + 1);
```

Bu program bloğunda i değişkeni 0 dan 99 'a kadar değerleri alıyor. Fakat ekran çıktımızda 1 den 100 'e kadar olan sayıları görüntülüyoruz. Üstad programcılar genellikle bu yöntemi kullanırlar. Daha akıllıca bir yazım şekli de şöyledir:

```
for (int i = 0; i < 100; Console.WriteLine(++i));
```

Şimdi de herbirinde 20 karakter olan 3 kolon halinde bir dizinin elemanlarını gösterelim.

```
for (int i = 0; i < sdizi.Length; i++)
    if (i % 3 == 2)
        Console.WriteLine("{0,-20}", sdizi[i]);
    else
        Console.Write("{0,-20}", sdizi[i]);
```

Döngü içerisinde tek bir ifade olduğundan süslü parantezleri kullanmadık. If cümlecisi, ilk değerin 3 'e bölümünden elde edilen kalanın 2 ye eşit olup olmadığını kontrol ediyor. İ değeri yalnız 2,5,8,11 olduğunda bir sonraki satıra geçiyor ve kolonlarımız oluşuyor.

Genellikle bir dizinin elemanlarını karşılaştırmak için for döngüsü kullanırız.

Örneğin 30 kişilik yakın arkadaşlarımızın isimlerinden oluşan bir dizimiz var. Kaç arkadaşımızın isminde E harfi yer aldığını hesaplamak istiyoruz:

```
char[] harfara = {'e', 'E'};
int eharfliler = 0;

for (int i = 0; i < isimler.Length; i++)
    if (isimler[i].IndexOfAny(harfara) != -1)
        eharfliler++;
```

Buradaki harfara dizisi char tipinde olup "E" ve "e" karakterlerini içermektedir. If bloğunda String sınıfının bir metodu olan IndexOfAny ile harfara dizisindeki elemanlar karşılaştırılıyor. Eğer isimde herhangi bir e karakteri yer almıyor ise IndexOfAny metodu -1 değerini döndürecektir. Eğer -1 değeri dönmez ise eharfliler değişkeni 1 arttırılıyor.

Eğer bu isimleri başka bir diziye kopyalamak isterseniz :

```
using System;

class harfarama
{
    static void Main()
    {
        string[] isimler =
        {
            "Ömer Faruk Demir", "Ali Can", "Elima Aydın", "Sefa Filiz", "Ebru Sevim"
        };
        int eharfliler = 0;
        char[] harfara = {'z', 'Z'};

        // Öncelikle "e" veya "E" harfi olanların sayısını buluyoruz

        for (int i = 0; i < isimler.Length; i++)
            if (isimler[i].IndexOfAny(harfara) != -1)
                eharfliler++;

        // yeni bir string dizi oluşturuyoruz

        string[] eisimler = new string[eharfliler];

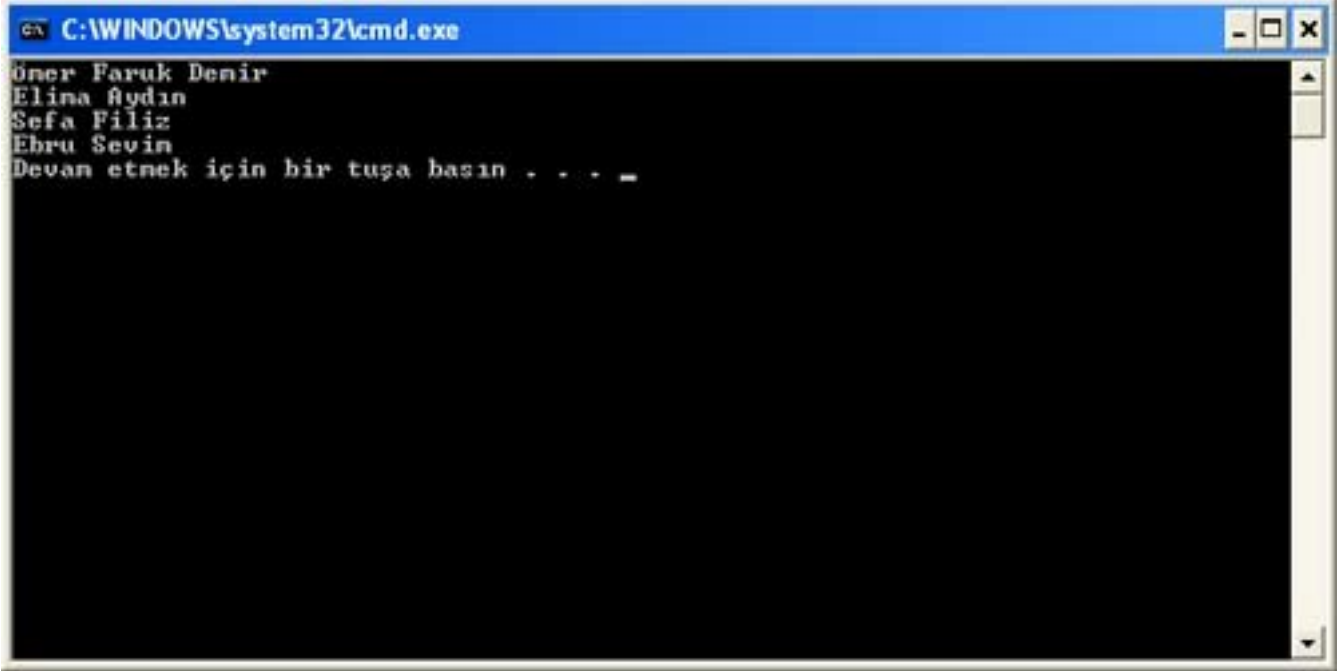
        // Bir diziden diğer diziye kopyalama işlemini yapıyoruz

        for (int i = 0, j = 0; i < isimler.Length; i++)
            if (isimler[i].IndexOfAny(harfara) != -1)
                eisimler[j++] = isimler[i];

        // Yeni oluşan dizimizi ekrana yazdırıyoruz

        for (int i = 0; i < eisimler.Length; i++)
            Console.WriteLine(eisimler[i]);
    }
}
```

Bu programda üç adet for döngüsü yer alıyor. İlk döngü yukarı da incelediğimiz aynısı, isimleri sayıyor. Program isminde E harfi olanların yer aldığı dizinin uzunluğunu hesaplıyor. İkinci döngü ile kopyalama işlemi gerçekleştiriliyor. Son döngü ile de yeni oluşan dizimiz ekrana yazdırmak için kullanılıyor.



Bir dizinin tüm elemanlarına erişmek için foreach döngüsü daha pratik olacaktır. Decimal tipte bir dizinin tüm elemanlarını foreach döngüsü kullanarak toplayalım:

```
decimal mtoplam = 0;  
  
foreach (decimal m in mdizi)  
{  
    mtoplam += m  
}
```

foreach cümlecüğünden sonra parantezler içerisine erişeceğimiz dizinin tipinde bir değişken tanımlıyoruz. Daha sonra dizimizin adını yazarak son elemana kadar döngünün sürdürülmesini sağlıyoruz. foreach döngüsünün bazı sınırlılıkları vardır. Örneğin döngü içerisinde dizinin herhangi bir elemanına ulaşamazsınız. foreach döngüsü yalnız döngülerde kullanılmaz. Örneğin string bir değişkenin elemanlarını (karakterlerini) ekrana yazdıralım:

```
foreach (char karakter in str)  
    Console.WriteLine(karakter);
```